VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Návrh informačního systému pro oční kliniku

Design of Information System for Ophthalmology

2018 Bc. Vavřinčíková Kateřina

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Kateřina Vavřinčíková**

Studijní program: N2649 Elektrotechnika

Studijní obor: 3901T009 Biomedicínské inženýrství

Téma: Návrh informačního systému pro oční kliniku
Design of Information System for Ophthalmology

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Analýza a popis současného stavu na oční klinice Fakultní nemocnice Ostrava.
2. Popis možností jazyka C# pro tvorbu PC aplikací.
3. Přehled databázových prostředků pro relační databáze.
4. Analýza požadavků a návrh celkové struktury informačního systému.
5. Návrh modulu administrace umožňujícího správu uživatelů a jejich oprávnění (rolí) a modulu správy pacientů umožňující zadávání pacientů pro informační systém oční kliniky.
6. Implementace modulu administrace a modulu správy pacientů v jazyce C# včetně komunikace s databází a začlenění do celkového informačního systému.
7. Testování funkčnosti modulů na oční klinice.
8. Zhodnocení výsledků práce a závěr.

Seznam doporučené odborné literatury:

[1] ROBINSON, Simon. *C#: programujeme profesionálně*. Vyd. 1. Brno: Computer Press, 2003. Programmer to programmer. ISBN 80-251-0085-5.
[2] NAGEL, Christian, Jay GLYNN a Morgan SKINNER. *Professional C# 5.0 and .NET 4.5.1*. Indianapolis, IN: John Wiley and Sons, 2014. ISBN 978-1-118-83294-3.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jaromír Konečný, Ph.D.**

Konzultant diplomové práce: doc. Ing. Martin Augustynek, Ph.D.

Datum zadání: 01.09.2016

Datum odevzdání: 30.04.2018

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení

„Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala."

V Ostravě dne 25. 4. 2018 Podpis: .....

Poděkování

Děkuji vedoucímu diplomové práce Ing. Jaromíru Konečnému, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a cenné rady při zpracování mé diplomové práce.

Abstrakt

Diplomová práce se zabývá návrhem informačního systému, dvou modulů a databáze, na základě požadavku Oční kliniky Fakultní nemocnice Ostrava.

Teoretická část práce zahrnuje popis současného stavu na oční klinice a popis jednotlivých ambulancí, dále popis jazyka C# a databázových prostředků pro tvorbu relační databáze.

Dále je práce zaměřena na praktický návrh této databáze a tvorbu dvou modulů. Modulu pro správu pacientů a administrátorského modulu. Je zde popsána tvorba těchto modulů, implementace a připojení k databázi. Závěrem byl systém otestován a nainstalován na oční kliniku.

Klíčová slova

Informační systém, oční klinika, Firebird, databáze, C#

Abstract

The diploma thesis is based on the design of an information system, two modules and a database, based on the request of the Ophthalmology Clinic of the Faculty Hospital in Ostrava.

The theoretical part of the thesis includes a description of the current state of the ophthalmology clinic and a description of individual ambulances, as well as a description of C # and database resources for relational database creation.

Further, the thesis is focused on the design of this database and the creation of two modules. Patient Management Module and Administrator Module. It describes the creation of these modules, implementation and connection to the database. Finally, the system was tested and installed at the eye clinic.

Key Words

Information system, Opthalmology clinic, Firebird, Database, C#

Obsah

Seznam použitých symbolů a zkratek

Seznam obrázků

Seznam tabulek

Úvod

Oční klinika FN Ostrava se zabývá všemi možnými poruchami vidění člověka. Rozděluje se na ambulance sloužící k diagnostice i k léčbě všech onemocnění oka. Diagnostických metod a aplikací na tomto pracovišti přibývá a tak byly stanoveny požadavky pro vznik databáze a softwaru, na který by se mohly ostatní moduly připojovat.

Cílem této diplomové práce je tedy navrhnout databázi a software, který urychlí a usnadní práci na této klinice. Informační system je rozdělen do dvou modulů. Modul pro správu pacientů a administrátorský modul.

V teoretické části je popsán současný stav na oční klinice, rozdělení jednotlivých ambulancí a popis jejich činností.

Dalším bodem v teoretické části je popis možností jazyka C# nejdříve v prostředí ASP .NET a poté v PC aplikacích. Je zde popsáno vývojové prostředí Visual Studio .NET, ve kterém byla vytvářena tato diplomová práce. Jsou zde popsány jednotlivé komponenty a možnosti práce v tomto prostředí a také jednotlivé struktury aplikace. Popis databázových prostředků je další teoretickou kapitolou, která je zde detailněji popsána. Můžeme zde nalézt základní pojmy týkající se relačních databází, ale také popis jazyka SQL včetně syntaxe jednotlivých příkazů.

Návrh databáze byl vytvořen na základě požadavků z FN Ostrava a databáze byla vytvořena v prostředí Firebird. Dále je popsán návrh obou modulů a jejich implementace v jazyce C#. Oba moduly vyžadují přihlášení uživatele. Jednotlivé uživatele může přidat administrátor v administrátorské aplikaci. Uživatel, který se poté přihlásí do aplikace pro správu pacientů bude mít možnost přidat nebo vyhledat nového pacienta a podívat se na historii jeho vyšetření.

Cílem této práce je tedy tvorba databáze a vytvoření dvou aplikací, které budou využívat uživatelé na oční klinice. Databáze je navržena tak, aby se na ni mohly připojit vytvořené aplikace. V současné době je již provázána se dvěmi dalšími aplikacemi, sloužícími pro vyšetření barvocitu a zrakové ostrosti pacienta. V modulu pro správu pacientů se pak může uživatel podívat jaká veškerá vyšetření pacient vykonal.

1 Popis současného stavu na oční klinice FNO

Oční klinika FNO je, jako jediné zařízení v Moravskoslezském kraji, akreditované pracoviště III. stupně. Nachází se v budově polikliniky FN. Oční klinika je rozdělena na několika ambulancí, které si popíšeme dále.

Na této oční klinice se nově příchozí pacient zapíše na recepci v čekárně. Poté je volán do ambulance na potřebná vyšetření. Podle diagnózy se stanoví typ vyšetření a daný přístroj na kterém se bude provádět. Výsledky tohoto vyšetření se vytisknou a jsou vloženy do dokumentace pacienta. Ten je poté zavolán do ordinace lékaře, který posoudí výsledky z daných vyšetření.

- Všeobecná oční ambulance

Tato ambulance slouží jako spádová pro více než 1,2 mil. obyvatel. Poskytuje jak preventivní, tak léčebnou péči. Docházejí zde pacienti pravidelně na běžné vyšetření zraku, pro předepsání brýlí, ale také pacienti s akutními či chronickými onemocněními oka.

- Vitreoretinální centrum

V této ambulanci se provádí diagnostika a léčba pacientů s onemocněním sítnice a sklivce. Jedno z hlavních onemocnění sítnice je diabetická retinopatie, a proto do této ambulance přichází na vyšetření především diabetici. Tato ambulance je vybavena fundus kamerou pro snímání očního pozadí, dále přístrojem pro OCT vyšetření, které zaznamená řez sítnice oka a dokáže odhalit makulární degeneraci, tedy postižení místa nejostřejšího vidění. Jako diagnostickou metodu zde můžeme uvést fluorescenční angiografii, která využívá prostupu kontrastní látky cévami, a tak zobrazuje kvalitu prokrvení cév v oku. Jako léčebnou metodu v této ambulanci můžeme zmínit laserovou koagulaci sítnice, která se využívá např. při diabetické retinopatii nebo při trhlinách v sítnici. Onemocnění sítnice se většinou zachytí při základním vyšetření ve spádové oční ambulanci a pacient je poté dlouhodobě sledován ve vitreoretinální ambulanci. Vyšetření zde probíhá pouze na základě objednání.

- Glaukomová poradna

Zelený zákal, tzv. glaukom, je onemocnění, při kterém dochází ke změnám zrakového nervu. Toto onemocnění probíhá dlouho bez příznaků, časem pacient může pozorovat, že dochází ke zužování zorného pole a bez léčby toto onemocnění může vést až ke slepotě. Jedním z rizikových faktorů této nemoci je zvyšování nitroočního tlaku. Ten je třeba pravidelně sledovat. Tato ambulance tedy poskytuje pacientům z široké spádové oblasti podrobné vyšetření glaukomu, a tak stanovení nejefektivnější léčby. Glaukomová poradna je vybavena především perimetrem, který zkoumá stav zorného pole pacienta a přístrojem pro měření nitroočního tlaku. Dalším důležitým přístrojem je pachymetr, který měří tloušťku rohovky a pomáhá lékařům upřesnit naměřené hodnoty nitroočního tlaku. Vyšetřuje se zde také přístrojem pentacam, který mapuje přední segment oka a oblast komorových úhlů, které také mohou zhodnotit přítomnost glaukomových změn.

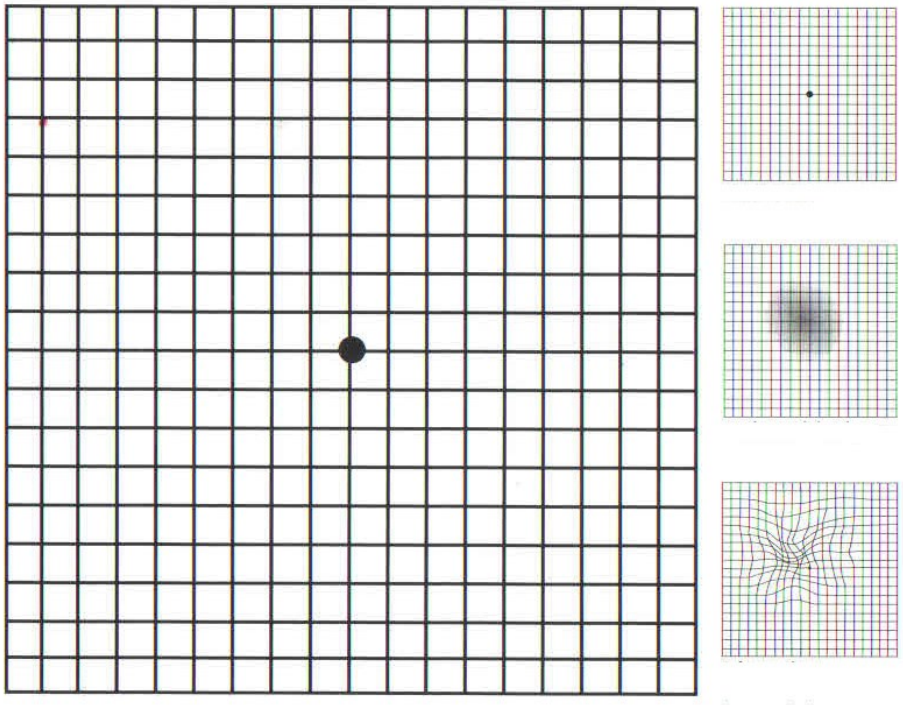
Obrázek 1: Zúžování zorného pole u glaukomatiků

- Centrum pro léčbu VPMD

VPMD, tedy věkem podmíněná makulární degenerace, je onemocnění makuly (centra nejostřejšího vidění). VPMD má dvě formy, suchou a vlhkou. Suchou formu má cca 85% lidí trpících VPMD, vlhká forma se objevuje u 15% nemocných, je však vážnější. Průběh vlhké formy VPMD je rapidní a ke ztrátě zraku může dojít již během několika měsíců. Hlavními příznaky vlhké formy je vlnění obrazu. Při včasné diagnostice se však tato forma dá léčit pomocí anti-VEGF terapie. Toto onemocnění se nejlépe ukáže při vyšetření OCT kde se skenuje řez makulou a je zde vidět výrazný otok sítnice.

Průběh suché formy VPMD je pomalejší a zrak se zhoršuje pozvolna během několika let. Příznakem je rozostřené vidění, především za tmy nebo v šeru. Příčinou je úbytek buněk citlivých na světlo. Léčba tohoto onemocnění bohužel neexistuje. Doporučují se výživové doplňky pro oči s obsahem Luteinu.

Příznaky jakékoliv formy VPMD pacient může odhalit již sám, pokud vlastní tzv. Amslerovu mřížku. Je to mřížka, která se pacientovi s vlhkou formou VPMD začne vlnit či křivit a pacientovi se suchou formou se rozostří či ztmavne. Na obrázku č.2 můžeme vidět klasickou mřížku.



Obrázek 2: Amslerova mřížka a její křivení

- Centrum pro děti s vadami zraku

Během prvních 12ti měsíců života prochází dětské oči největšími změni, a okolo 8 roku života teprve dítě vidí jako dospělý. Proto může docházet k narušení vývoje očí a mohou vznikat oční vady, které jsou při včasné léčbě vyléčitelné. K nejčastějším vadám zraku patří krátkozrakost, dalekozrakost, astigmatizmus, tupozrakost nebo šilhání. V této ambulanci se lékaři specializují především na diagnostiku těchto zrakových vad a jejich léčbě.

2 Popis možností jazyka C# pro tvorbu PC aplikací

Jazyk C# byl navržen pro použití v prostředí .NET Framework. Tato platforma slouží pro vývoj, šíření a spouštění aplikací. Pracovní prostředí .NET Framework využívá prostředí zpracování, které se označuje zkratkou CLR nebo operační prostředí .NET. Kód, který se spouští pod kontrolou operačního prostředí .NET se nazývá spravovaný kód.

Pokud budeme chtít kód vyvinutý v jazyce C# spustit v prostředí .NET je zapotřebí jej zkompilovat (přeložit). Nejprve se kód přeloží do jazyka MSIL a následně se tento kód přeloží modulem CLR na kód využívaný v dané platformě.

2.1 ASP.NET

ASP je technologie firmy Microsoft. Tato technologie vývojářům umožňuje vytvářet webové stránky s dynamickým obsahem. Jedná se tedy o soubor HTML vytvořený v jazyce VBScript nebo JavaScript. Novější revize technologie ASP je ASP.NET, která obsahuje opravy mnoha zjištěných problémů a nevýhod. Jednou z výhod je také, že aplikace ASP.NET se mohou programovat v jazyce C#. Další výhodou je možnost práce v jednotném vývojovém prostředí. V prostředí Visual Studio .NET se mohou tvořit samotné stránky, ale také programovat přístup k datům. Technologie ASP.NET se používá také pro tvorbu stránek, které vyžadují zobrazení formulářů v prohlížeči. Tato výhoda se především uplatňuje v aplikacích, které se spouští na velkém počtu počítačů a potřebují tedy bohatší uživatelské rozhraní.

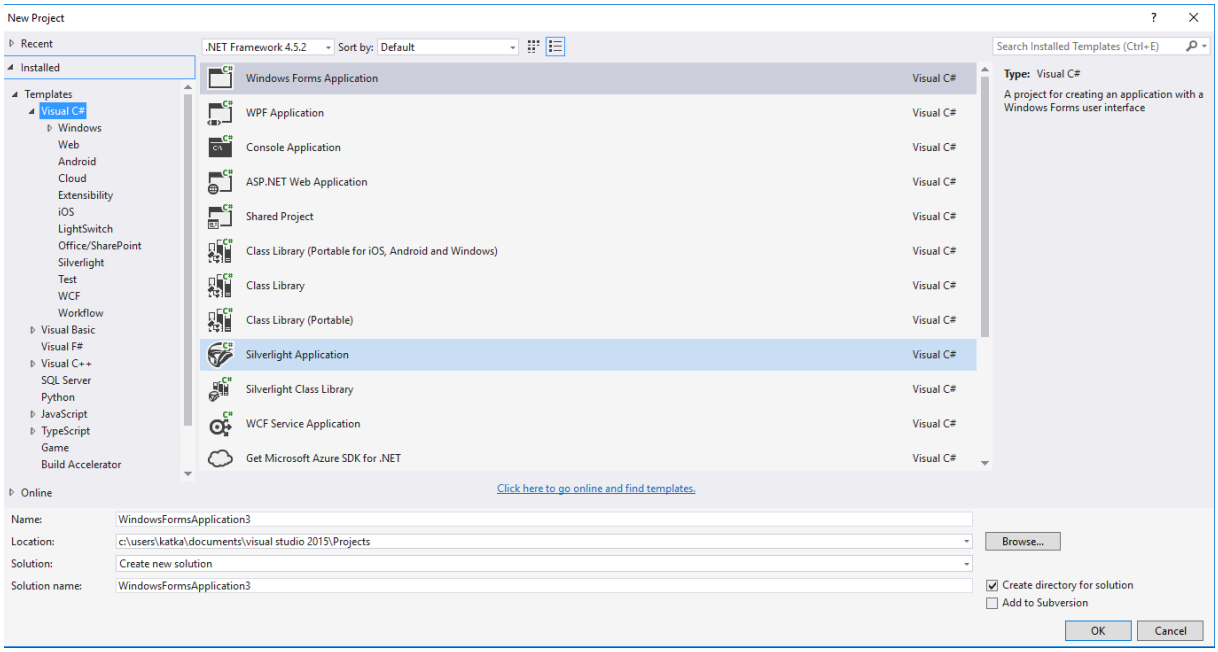
Tento framework využívá architektury klient-server. Vytvořené webové stránky tak běží na serveru, klient vyšle požadavek na server a ten mu spustí požadovanou webovou stránku.
Podobnost s vývojem Windows aplikací je v použitých komponentách. V ASP.NET se nazývají *serverové komponenty* a může se jednat o podobné komponenty, které se využívají při tvorbě desktopových aplikací. Jedná se o grafické komponenty, jakými jsou *TextBoxy, Tlačítka, GridView* a další. Na pozadí tohoto grafického rozhraní máme kód, do kterého vkládáme jednotlivé funkcionality, metody a události jednotlivých komponent.

- Rozdíl mezi aplikacemi pro web a pro Desktop

Desktopové aplikace se v současné době vyvíjejí velmi snadno. Máme k dispozici připravené knihovny či prvky panelu Toolbox. Z těchto prvků jsou sestavovány formuláře a tlačítky jsou propojeny. Je zde připraven seznam dostupných ovládacích prvků nebo možnost vytvořit si vlastní ovládací prvky dle potřeb vývojáře. Jedná se o objektové modely jejichž vzhled a funkce jsou ovlivňovány vlastnosti daného prvku. Je možnost upravovat funkce tlačítek či události vyvolané uživatelem. Tento vývoj je pro vývojáře velmi pohodlný a intuitivní. Tyto aplikace umožňují uložení dat na lokálním počítači či na serveru. Nevýhodou pro správce je pracnost podpory či aktualizací desktopových aplikací. To vedlo k velkému rozmachu webových aplikací. Jejich správa je značně jednodušší, jelikož klient k užívání určité aplikace potřebuje pouze počítač s podporou HTML. Nevýhoda těchto aplikací spočívá ve velkých nákladech a pracnosti jejich vývoje. HTML podporuje užívání pouze jednoduchých ovládacích prvků, ke tvořbě složitějších prvků je zapotřebí velké znalosti programátora. To vyústuje ve velmi nečitelný a složitý kód, jehož změna či údržba je velmi náročná. [10]

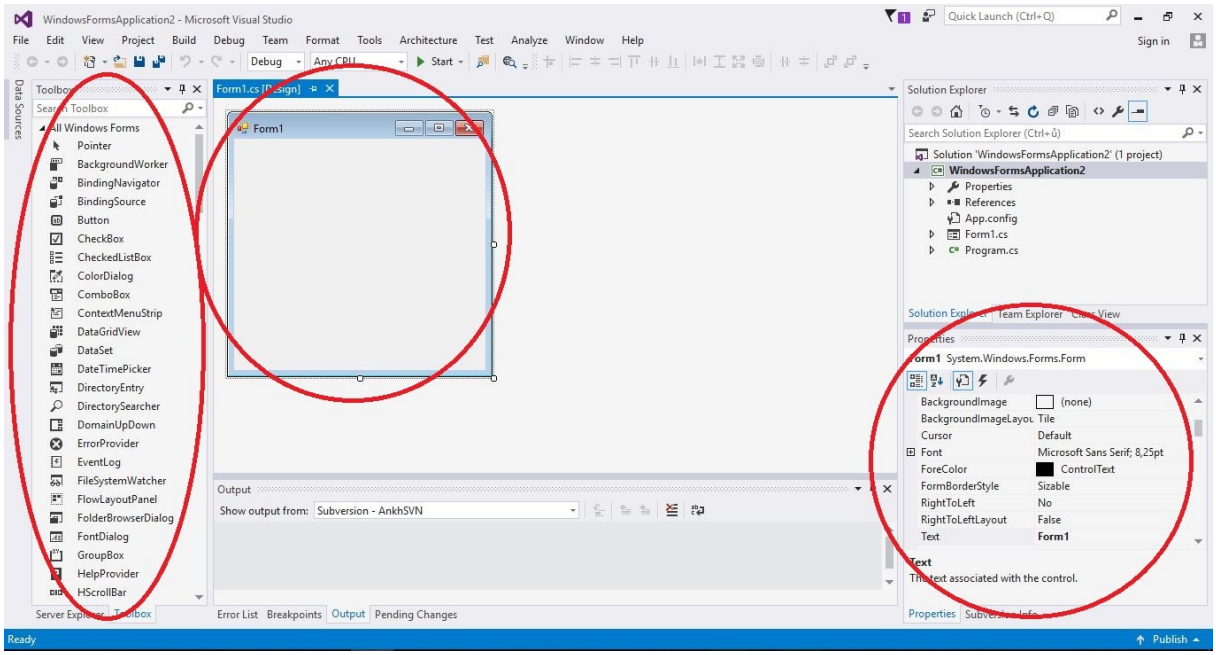
2.2 Jazyk C# pro tvorbu Windows aplikací

Jazyk C# se mimo programování webových aplikací používá v prostředí .NET Framework také ke tvorbě desktopových aplikací na osobní počítače. Pro tvorbu těchto aplikací je nejpoužívanějším prostředím Visual Studio .NET. Program Microsoft Visual Studio byl využit při tvorbě této diplomové práce. K vytváření těchto aplikací v programu Microsoft Visual Studio se používají modely Windows Form. Aplikaci Windows si vývojař vytvoří v programu stisknutím tlačítka File → New → Project. V následujícím dialogovém okně si zvolí adresář Visual C#, kde je na výběr několik typů projektů. Pro tvorba základní Windows aplikace si vývojař zvolí Windows Form Application.



Obrázek 3: Vytvoření nové aplikace Windows v jazyce C#

Následně Visual Studio .NET vytvoří adresář a počáteční soubory aplikace a zobrazí se první prázdný formulář, který si vývojář nadále upravuje. Požadované formuláře si vývojář sestaví pomocí ovládacích prvků, které jsou dostupné v panelu nástrojů v programu. Poté se naprogramují jednotlivé ovládací prvky a události okna dle potřeby.



Obrázek 4: Prázdný formulář nového projektu Windows Form

- **Panel Toolbox**

Panel Toolbox obsahuje řídící prvky, kterými se vytváří požadované grafické prostředí formuláře Windows Form. Díky těmto prvkům můžeme používat textová okna, tlačítka, nabídky apod. V tabulce je uveden seznam nejčastěji využívaných prvků aplikace v této diplomové práci.

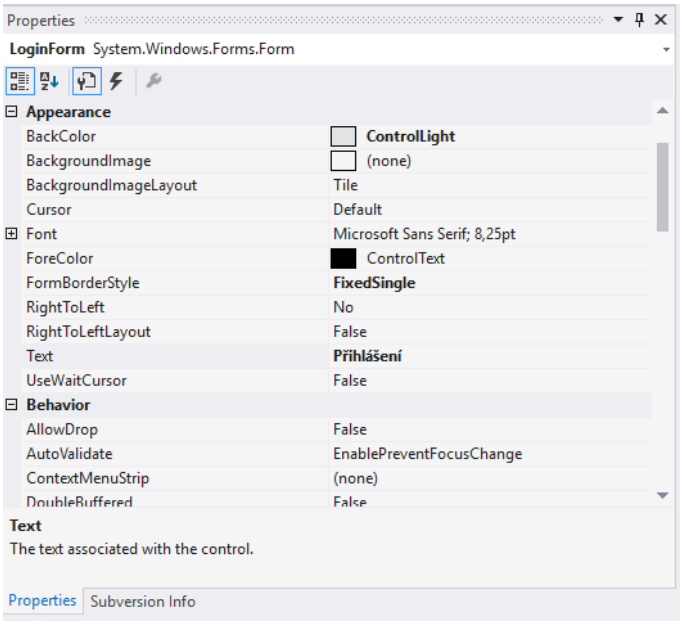
Tabulka 1: Využívané prvky Windows Form aplikace

Prvek	Popis
Label	Návěstí - zobrazuje text
Button	Tlačítko
TextBox	Textový rámeček
CheckBox	Zaškrtávací pole
PictureBox	Rámeček s obrázkem
DataGridView	Datová oblast
ListBox	Seznam
BindingSource	Datový zdroj komponent
DataSet	Datový zdroj
ListView	Komponenta pro zobrazování kolekcí
ProgressBar	Zobrazuje stav příslušné operace v procentech
PropertyGrid	Tabulka vlastností

- **Panel Properties**

Tento panel uvádí vlastnosti daného ovládacího prvku, které si můžeme nastavit. Je zde uvedena například barva pozadí daného prvku, font písma, ale také ukotvení prvku ve formuláři či jeho název a

další. Okno Properties se zobrazuje v pravém dolním rohu programu, jak lze vidět na obrázku č.4.



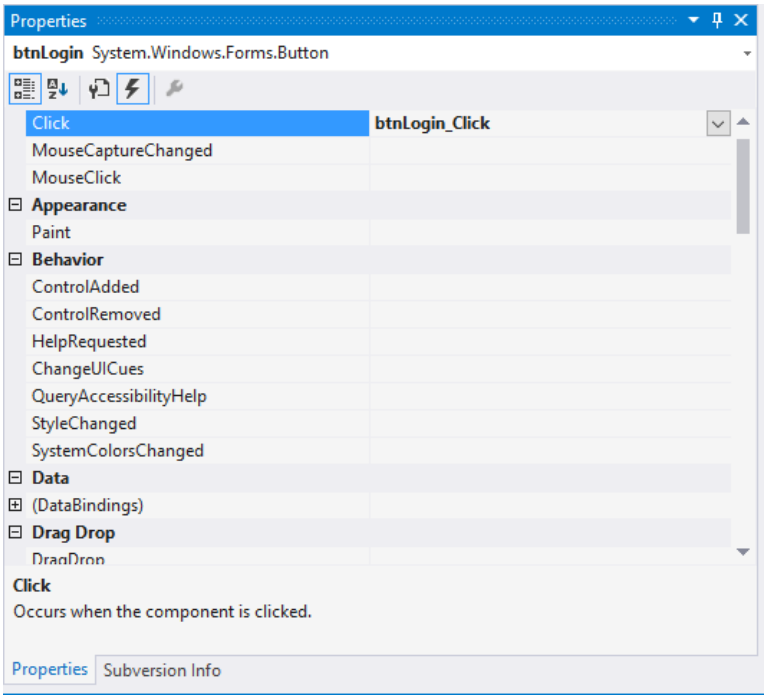
Obrázek 5: Okno Properties

- **Panel Designer**

V tomto panelu můžeme vidět požadovaný formulář a můžeme do něj přidávat prvky z panelu Toolbox. Na obrázku č.4 vidíme formulář zatím prázdný.

- **Události**

Ve Windows Form aplikacích jsou jednou z důležitých věcí události. Pro správnou komunikaci s uživatelem a obsluhu aplikace je třeba události využívat. Jsou dostupné v panelu Properties pod tlačítkem se symbolem blesku. Zde jsou uvedeny události, které se vztahují k danému objektu nebo ovládacímu prvku. Kliknutím na událost se nám do kódu třídy vloží nová metoda a otevře se editor kódu.



Obrázek 6: Události v panelu Properties

2.2.1 Struktura aplikace

Doposud jsme se zabývali pouze grafickým prostředím aplikace, ale Visual Studio .NET zobrazuje buď GUI aplikace nebo jeho zdrojový kód. Ke kódu se dostaneme pravým kliknutím na grafický formulář a stisknutím možnosti View Code. Daný formulář je reprezentován třídou (výchozí pojmenování Form1.cs), která obsahuje pouze konstruktor InitializeComponent(). Tento konstruktor slouží k tomu, aby se otevřel daný formulář a v něm prvky, které jsme si zvolili i s jejich vlastnostmi. Třídy jsou ve struktuře celé aplikace důležitou součástí, a tak se jimi budeme dále zabývat.

- **Třídy**

Třída je vzor, podle kterého se vytváří objekty, jejich vlastnosti a schopnosti. Data a chování jsou členy třídy, a zahrnují její metody, vlastnosti, události apod. Data a funkce uvnitř třídy se nazývají **členy**. Tyto členy můžeme definovat jako **veřejné** (public) nebo **soukromé** (private). Veřejný člen je dostupný i pro jiné třídy, zatímco soukromý je využitelný pouze uvnitř dané třídy. **Datové členy** nám ukládají data třídy, jako jsou složky, konstanty a události. Mezi **funkční členy** patří např. metody, vlastnosti, konstruktory či operátory. Poskytují nám tedy funkce pro manipulaci s daty dané třídy.

Příklad kódu pro prázdný formulář:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Namespace nám v kódu označuje tzv. jmenný prostor, do kterého se sdružují třídy.
Using nám umožňuje používat třídy jiných jmenných prostorů.

- **Metody**

Metody jsou deklarovány v třídě nebo ve struktuře. Je to blok kódu, který obsahuje sled příkazů, které lze volat z jiných částí programu. Metody se deklarují následujícím způsobem:

```
[přístup] [návratový typ] [jméno metody]([parametry])
```

Přístup volíme tedy *public* či *private.* Pokud volíme public, metoda bude veřejná. Pokud modifikátor nepoužijeme VS .NET bude brát metodu jako soukromou, tedy private. Pokud zvolíme návratový typ *void*, nebude metoda vracet žádnou hodnotu. Pokud metoda hodnotu vracet bude, musí být v těle metody použit příkaz *return*. Následuje samotné jméno metody. Závorka s parametry je povinná, avšak pokud metoda žádné parametry nemá, necháváme závorku prázdnou. Parametr se skládá z datového typu a názvu, na který se poté můžeme odkazovat.

Příklad:

```
class Hello
{
        public void Pozdrav()
        {
                Console.WriteLine("Hello world!");
        }
}
```

- **Pole**

Použití polí v jazyce C# je velmi užitečné, pokud si vývojář přeje pracovat a manipulovat s více daty. Pole je tedy struktura dat či prvků, mající stejný datový typ, vlastnosti a metody. Pro definici pole používáme hranaté závorky, které umístíme za datový typ obsažených prvků. Za rovnítkem je třeba pole inicializovat. Použíjeme klíčové slovo new a za něj datový typ a popř. velikost pole.

```
Datovy_typ[] pole = new datovy_typ[počet prvků];
```

Příklad:

```
int[] pole = new int[50];
```

2.2.2 Datové typy

Datové typy nám určují, co můžeme s hodnotami uloženými do proměnných provádět a jak se budou hodnoty ukládat. V následující tabulce jsou uvedeny předdefinované datové typy v jazyce C#. [2] [4] [7] [8]

Tabulka 2: Předdefinované datové typy

Předdefinované typy	Typ	Popis	Velikost	Typ v .NET Framework
Celočíselné typy	Sbyte	znaménkové celé číslo	8 bitů	System.Sbyte
	Byte	neznaménkové celé číslo	8 bitů	System.Byte
	Short	znaménkové celé číslo	16 bitů	System.Int16
	Ushort	neznaménkové celé číslo	16 bitů	System.UInt16
	Int	znaménkové celé číslo	32 bitů	System.Int32
	Uint	neznaménkové celé číslo	32 bitů	System.UInt32
	Long	znaménkové celé číslo	64 bitů	System.Int64
	Ulong	neznaménkové celé číslo	64 bitů	System.UInt64
Typy s desetinnou čárkou	Float	číslo s desetinnou čárkou	7 čísel	System.Single
	double	číslo s desetinnou čárkou	15-16 čísel	System.Double
	decimal	číslo s desetinnou čárkou	28-29 čísel	System.Decimal
Booleovský typ	Bool	bitové číslo (true nebo false)	8 bitů	System.Boolean
Znakový typ	Char	unicode znaky	16 bitů	System.Char
Odkazové typy	Object	kořenový typ		System.Object
	String	řetězec znaků		System.String

3 Přehled databázových prostředků pro relační databáze

Relační databáze můžeme považovat za tabulkovou strukturu informací, pro které existují různé metody přístupu k datům. Data jsou uchovávána v indexovaných sekvenčních souborech, kde každá tabulka obsahuje jedinečný identifikátor tzv. **primární klíč**. Cílem modelování dat je popsat model systému, který je pak použit k vytvoření databáze obsahující stejné informace, jako původní systém Datové modely mají zásadní význam pro vývoj informačních systémů. Zachycují dynamické a statické vlastnosti, potřebné pro podporu požadovaného procesu. Konceptuální modelování je proces modelování všech vlastností aplikace, a je povinen umožnit rozvoj datových modelů pro konkrétní aplikace. Datové modelování vytváří schéma, které definuje objekty, atributy a vztahy.

Při návrhu databáze je nutné definovat všechny entitiy v systému a vztahy mezi nimi. Existují tři možné typy vztahů mezi entitami:

- 1:1 – nejjednodušší vztah mezi dvěma subjekty. Jedné položce v první tabulce odpovídá jedna položka ve druhé tabulce.
- 1:M – jedné položce v první tabulce odpovídá N-položek ve druhé tabulce.
- M:N – M-položkám v první tabulce odpovídá N-položek ve druhé tabulce.

Poslední pojem terminologie modelování dat je atribut. Atributy jsou charakteristické vlastnosti, které se týkají každého subjektu. Může se jednat o atributy např. jméno, adresa, datum narození atd. [1]

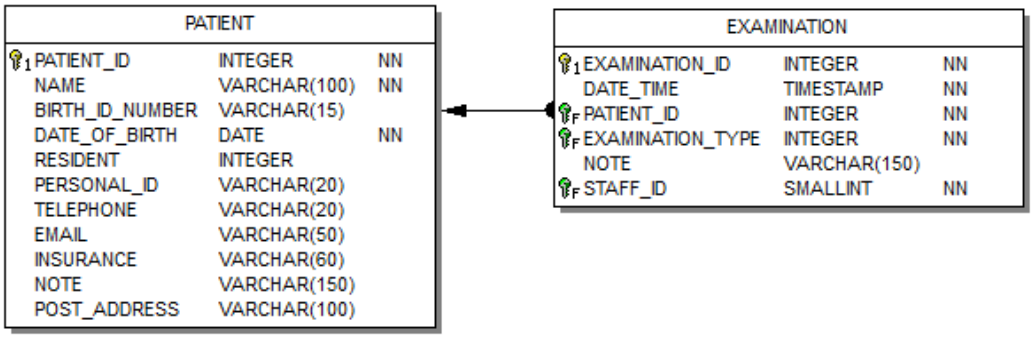
3.1 Základní pojmy

Tabulka je základním stavebním prvkem databáze. Skládá se ze sloupců a řádků, které ji definují. Sloupce odpovídají jednotlivým atributům a řádky jednotlivým prvkům. Soubor všech potřebných tabulek nám tedy dává celou databázi. Tabulka popisuje jednu entitu a atributy popisují vlastnosti, které si chceme o daném objektu ukládat. Seskupení všech tabulek a tedy celé databaáze se nazývá **relační datový model** (RD model).

Jednotlivé sloupce musí mít svůj předdefinovaný datový typ. Tabulka musí obsahovat alespoň jeden atribut, který bude jedinečným identifikátorem každého objetktu. Tento primární klíč je tedy unikátní a žádné dva řádky nesmí mít tento klíč shodný. Můžeme definovat take cizí klíče, které nám určují vazby mezi tabulkami, jedná se tedy o primární klíč z jiné tabulky. Každá tabulka má svůj unikátní název, který vystihuje její obsah.

EXAMINATION_ID	DATE_TIME	PATIENT_ID	EXAMINATION_TYPE	NOTE	STAFF_ID
1	05.04.2017 13:16	1	1	123	2
2	10.04.2017 07:17	1	1	123	3
3	10.04.2017 09:07	2	1	123	3
4	12.04.2017 06:49	3	1	123	3
5	12.04.2017 07:11	3	1	<null>	3
6	12.04.2017 07:26	4	1	<null>	3
7	12.04.2017 07:30	4	1	<null>	3
8	12.04.2017 08:23	5	1	<null>	3
9	12.04.2017 08:26	5	1	<null>	3
10	13.04.2017 05:35	6	1	<null>	3

Obrázek 7: Struktura relační tabulky Examination



Obrázek 8: Definice dvou tabulek a jejich klíčů

Na uvedeném obrázku můžeme vidět vytvoření tabulky *Patient* a tabulky *Examination*. V tabulce Patient jsme vytvořili primární klíč *Patient_ID* a ten je pak použit jako cizí klíč v tabulce *Examination*, tedy vyšetření. V tabulce vyšetření máme opět primární klíč *Examination_ID*. Tento indentifikátor je vždy datového typu *int* a nesmí být *NULL*.

3.2 Jazyk SQL

Relační databázové systémy jsou postaveny na základech jazyka SQL. Slouží ke správě, organizování a získávání dat z databáze. Dovoluje nám také např. upravovat strukturu databáze, zabezpečení, oprávnění uživatelů, tvořit dotazy apod. Jedná se o jazyk neprocedurální, což znamená, že kód jazyka SQL nepíšeme v žádném program, ale vkládáme jej do jiného programovacího jazyka.

Hlavními možnostmi, jak využívat jazyk SQL je:

- Definice dat
- Získávání dat
- Manipulace s daty
- Řízení přístupu
- Sdílení dat
- Integrita dat

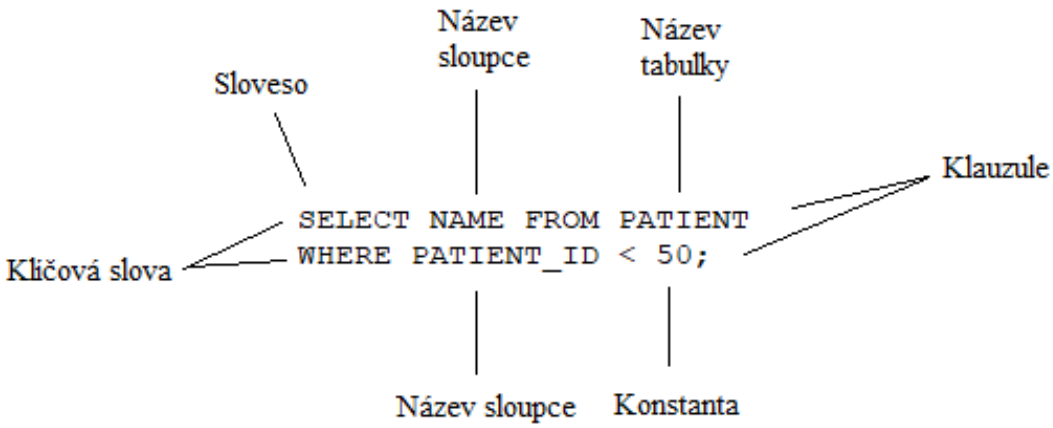
Možnosti užívání se liší podle toho, kdo s nimi pracuje. Část je určená pro administrátory, a část pro koncové uživatele a programátory. První částí je jazyk DDL , což je jazyk pro tvorbu databázových schémat. Způsob jejich ukládání řeší jazyk SDL. Pro návrháře existuje jazyk VDL, který určuje vytváření pohledů. A poslední a nejdůležitější částí je jazyk DML, který definuje základní příkazy. V následující tabulce jsou uvedeny nejčastěji užívané SQL příkazy:

Tabulka 3: Hlavní příkazy jazyka SQL

	Příkaz	Popis
Manipulace s daty	SELECT	Získává data z databáze
	INSERT	Vkládá data do databáze
	UPDATE	Upravuje stávající data
	DELETE	Odstraňuje data z databáze
Definice dat	CREATE	Vkládá do databáze např. tabulku, index, schéma, doménu
	DROP	Odstraňuje z databáze např. tabulku, index, schéma, doménu
	ALTER	Mění v databázi např. tabulku, index, schéma, doménu
Řízení přístupu	GRANT	Přiděluje uživateli daná práva
	REVOKE	Odebírá uživateli práva
Řízení transakcí	COMMIT	Ukončuje a potvrzuje aktuální transakci
	ROLLBACK	Ruší aktuální transakci
Programové SQL	OPEN	Otevírá tabulku s výsledky dotazu
	CLOSE	Zavírá tabulku
	DESCRIBE	Přidává do dotazu popis

- **Syntaxe jazyka SQL**

Jazyk SQL se řídí standardy ANSI/ISO, které mimo jiné specifikují klíčová slova, které se využívají v klauzulích příkazu. Každý příkaz tedy začíná slovesem z předchozí tabulky a dalšími klauzulemi, které specifikují data, s nimiž chceme pracovat nebo poskytují informace o tom, co by měl příkaz udělat. Tyto klauzule začínají klíčovým slovem, jako je např. *WHERE, FROM, INTO.* [3] [4]



Obrázek 9: Příklad struktury příkazu pro získání jmen pacientů

3.3 Databázové řídící systémy

Systém, který spravuje informace v databázi a v počítači je nazýván databázovým řídícím systémem. Řídícím databázovým systémem je v případě elektronické databáze software, který uchovává a zpracovává soubory a data na disku a v paměti počítače. Takovými řídícími softwary může být např. MySQL, Oracle, Firebird nebo PostgreSQL.

- **MySQL**

MySQL je světově nejpopulárnější open source databáze. Jedná se o vylepšení již existující databáze mSQL, která zaručuje větší flexibilitu, rychlost, a umožňuje použít vlastní modul pro ukládání dat. Je to nejpoužívanější databáze pro webové aplikace.

- **Oracle**

Oracle je velice složitý a výkonný databázový system, který strukturovaně spravuje data. Používá se pro správu dat ve velkých a také státních firmách. Ve verzi Express není nutná placená licence.

- **PostgreSQL**

Jedná se o následovníka POSTGRES. Je to open source objektově-relační databázový systém. Má vice než 15 let aktivního vývoje a osvědčené architektury, která si získala silnou pověst pro svou spolehlivost, integritu dat a správnost. Systém byl navržen v roce 1986 Michaelem Stonebrakerem z University of California. Tento databázový system je použitelný pod operačními systémy Linux, UNIX i Windows. Systém má plnou podporu cizích klíčů, podporuje transakční zpracovávání dotazů a zaručuje konzistenci dat v databázi. Jedná se především o databázi využívanou pro webové aplikace. [5]

- **Firebird**

Je odvozen ze zdrojového kódu Borland InterBase 6.0. Firebird vznikl v roce 2000 a od té doby pokračuje pod vedením neziskové organizace Firebird Foundation. Firebird je relační databázový system nabízející mnoho vlastností dle ANSI SQL standardů použitelných pod systémy Linux, UNIX a Windows. Nabízí plnou podporu ACID transakcí, uložených procedur a spouští podporu externích funkcí a velké množství nástrojů třetích stran obsahujících grafické administrační nástroje. Je zde velké množství způsobů jak přistupovat k databázi: API, dbExpress ovladače, ODBC, .NET provider, Python modul, PHP.

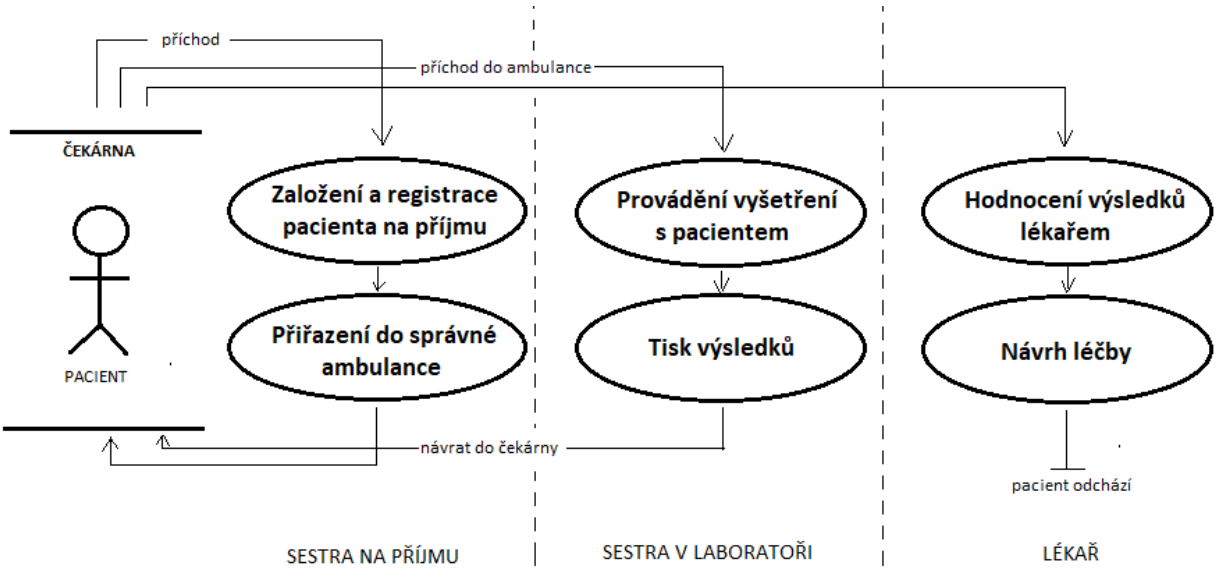
Výhodou tohoto systému je vysoká propustnost paralelního zpracování transakcí, okamžité zotavení po pádu a menší nároky na zdroje systému. Tento přístup později převzala databáze PostgreSQL. Další výhodou tohoto relačního systému je možnost pracovat s databázemi o velikosti desítek terabajtů a možnost uchovávat vice než dvě miliardy řádků v každé tabulce. Díky možnosti uložení velkého množství dat jsem se pro tvorbu databáze rozhodla využít tento relační systém. Dalším rozhodovacím kritériem byla možnost využívání databáze současně z více klientských aplikací, což systém Firebird splňuje.

Tento relační system se skládá z databázového serveru provozovaného na síti a klientské knihovny .DLL. Základním protokolem pro komunikaci klient-server je protocol TCP/IP.

Při tvorbě databáze byl použit ovladač Firebird .NET Data Provider, což je open source ovladač určený pro použití v .NET aplikacích. Ovladač je vytvořen v jazyce C#. Pro přehlednější tvorbu a práci s databází můžeme využít grafické administrační nástroje. V této diplomové práci byl použit nástroj IBExpert. [6]

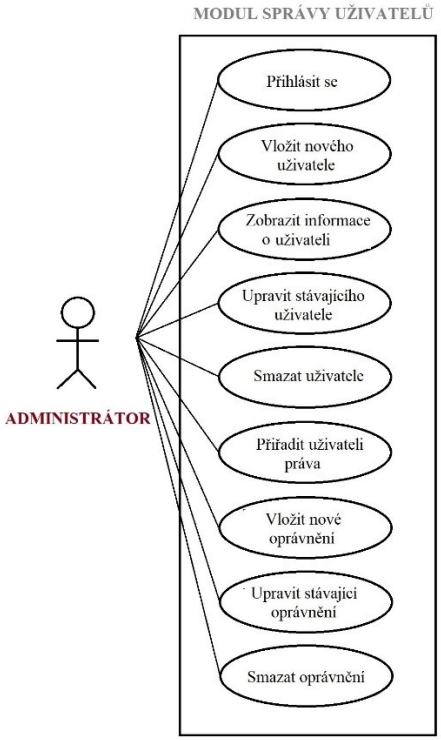
4 Analýza požadavků a návrh celkové struktury informačního systému

Požadavkem kladeným oční klinikou FN Ostrava bylo vytvořit systém, který by umožňoval správu uživatelů a jejich práv a také správu pacientů. U správy pacientů jde především o to, aby měl lékař ucelený pohled na všechna vyšetření, které pacient absolvoval. Celý systém by měl být navržen tak, aby bylo možné kdykoliv přidat nové komponenty a aplikace. Proto bylo také úkolem této diplomové práce realizovat databázi, která by zahrnovala výše zmíněné moduly. Před samotným návrhem databáze byl pečlivě prostudován chod oční kliniky a byly zjištěny informace, které je třeba v databázi u pacientů a uživatelů evidovat. Z následujícího diagramu lze vidět, jak to současně funguje na oční klinice. Nově příchozí pacient je registrován sestrou na příjmu, která rozhodne, do které ambulance pacient patří. Až bude pacient v ambulanci na řadě, bude sestrou zavolán, a v ambulanci mu budou provedena potřebná vyšetření, sestra vyšetření vytiskne, založí ke kartě pacienta a ten se vrací zpět do čekárny. Poté je pacient volán k lékaři, který dle vyšetření posoudí oční nález pacienta. Nově vytvořený informační systém má pomoci lékaři efektivněji hodnotit vyšetření, která se mu zobrazí na počítači. Nedojde tak ke zkreslení výsledků, např. špatným tiskem tiskárny.



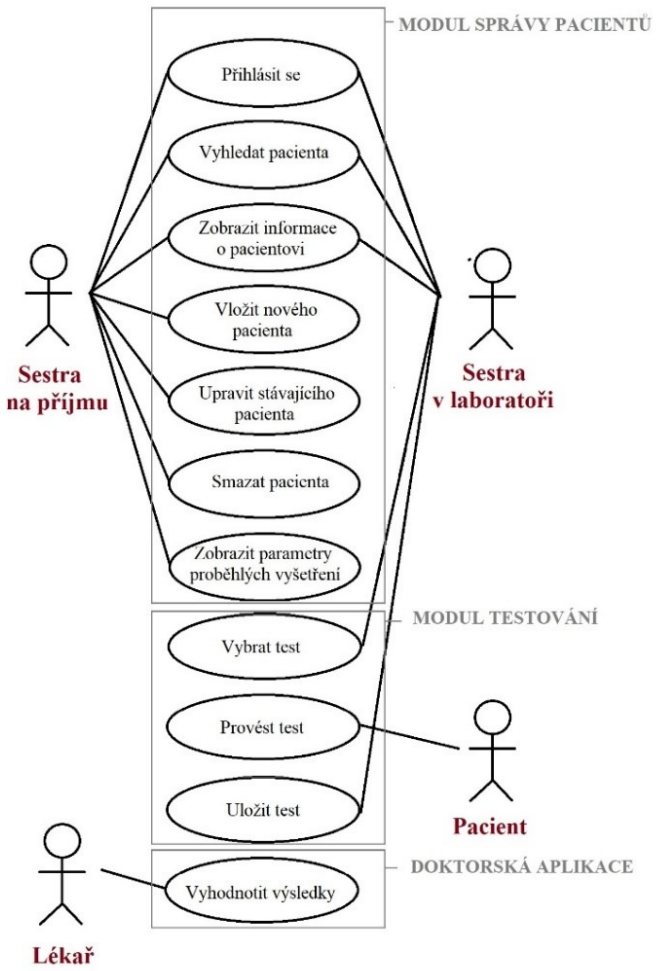
Obrázek 10: Současný stav na oční klinice z hlediska pacienta

Úkolem diplomové práce je tedy především databáze pacientů a poté modul správy pacientů a modul administrátora. Sestra na příjmu bude mít k dispozici modul správy pacientů, zde zapíše pacienta při jeho příchodu. Poté si sestra v laboratoři pacienta vyhledá a provede s ním vyšetření. A nakonec si lékař z databáze najde pacienta a požadovaná vyšetření si otevře. Jednotlivé aplikace pro vyšetření a doktorská aplikace jsou součástí projektu pro tuto kliniku a vytvářejí je další studenti.



Obrázek 11: UseCase diagram administrátorského modulu

UseCase diagram popisuje činnosti jednotlivých uživatelů v každém modulu. V modulu pro správu pacientů by měla být možnost přihlášení každého uživatele, vyhledání pacienta či úpravu jeho údajů a zobrazení informací o pacientovi, kde by měly být zobrazena všechna vyšetření, která daný pacient provedl a jméno uživatele, který s pacientem vyšetřením prováděl. Modul administrace by měl mít za úkol přihlášení administrátora, poté vložení a úpravu uživatelů, včetně stanovení jejich uživatelských jmen a hesel, a dále správu oprávnění vstupu do jednotlivých aplikací.



Obrázek 12: UseCase diagram modulu správy pacientů a dalších aplikací

Z UseCase diagramu modulu správy pacientů je patrné, že sestra na příjmu a sestra v laboratoři nebudou mít stejná práva. Nový pacient, který přijde na oční kliniku za účelem vyšetření probíhajíciho na počítači, bude nejprve založen sestrou na příjmu, která vyplní veškeré údaje o pacientovi. Pacient postupuje do vyšetřovny, kde ho daným vyšetřením bude provádět sestra v laboratoři. Ta si již vytvořeného pacienta vyhledá v databázi v již konkrétní aplikaci.

Souběžně s touto diplomovou prací se vytvářel také modul pro testování barvocitu, který vytvořila Ing. Monika Borová. Byla zde potřeba spolupráce při propojení databáze a testování funkčností. Při této spolupráci také s Ing. Jaromírem Konečným, Ph.D. bylo využito verzování zdrojových kódů pomocí systému SVN. Tato správa verzí nám pomohla v orientaci v proběhlých změnách v každé aplikaci, v návratu k předchozím verzím a ve sdílení databáze.

5 Návrh modulů administrace a správy pacientů

Prvním návrhem celkového systému byl návrh databáze. Ten byl sestaven dle vytvořených UseCase diagramů a podle požadavků na jednotlivé moduly. Databáze byla navržena a konzultována s personálem oční kliniky FN v Ostravě. Obsahuje také informace, které je třeba uchovávat při vyšetření barvocitu a zrakové ostrosti. Tyto aplikace vznikaly souběžně s databází a připojují se k ní.

5.1 Návrh databáze

Byla navržena a vytvořena databáze Firebird. Bylo nutné nainstalovat si Firebird server a pro lepší tvorbu a přístup k databázi bylo využito vizuálního prostředí IbExpert.

Při tvorbě databáze bylo využito pravidel konceptuálního modelování. Základním kamenen databáze byly požadavky personálu na Oční klinice. Z těchto požadavků byly vytvořeny základní entity reprezentující požadovanou databázi a z nich byl následně vytvořen konceptuální model, tzv. ERD schéma.

ERD model definuje určité entity a vztahy mezi nimi. Entita je reálný objekt množiny. V tomto případě se může jednat o určitého pacienta s reálným identifikačním číslem. Dalším důležitým prvkem je atribut, což je informace, kterou chceme u jednotlivých entit uchovávat. U pacienta je atributem např. jméno, rodné číslo atd. Je dobré si u každé množiny entit stanovit její primární klíč.

V následujících tabulkách jsou uvedeny jednotlivé entity, které je třeba evidovat v databázi. Databáze byla navržena a provázána tak, aby bylo možné ji postupně rozšiřovat a navázat na nově vznikající aplikace. Dále tabulky obsahují datové typy a jejich velikosti, obsahují také údaj, zda se jedná o primární či cizí klíč, zda může daný atribut nabývat hodnoty *null*, zda obsahuje index, a na závěr popis daného atributu.

Tabulka 4: Pacient

Patient						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Patient_ID	Integer	-	PK	Ne	Ano	ID pacienta
Name	Varchar	100	Ne	Ne	Ne	Jméno a příjmení pacienta
Birth_ID_number	Varchar	15	Ne	Ano	Ne	Rodné číslo
Date_of_birth	Date	-	Ne	Ne	Ne	Datum narození
Resident	Integer	-	Ne	Ano	Ne	Občan ČR (Ano/ne)
Personal_ID	Varchar	20	Ne	Ano	Ne	Číslo OP
Telephone	Varchar	20	Ne	Ano	Ne	Telefon
Email	Varchar	50	Ne	Ano	Ne	E-mail
Insurance	Varchar	60	Ne	Ano	Ne	Pojišťovna
Note	Varchar	150	Ne	Ano	Ne	Poznámka
Post_adress	Varchar	100	Ne	Ano	Ne	Adresa

V tabulce Pacient jsou uvedeny osobní informace o každém pacientovi. Nejsou zde uvedeny žádné informace o tom, která vyšetření pacient provedl, či do které ambulance se dostavil. Informace o vyšetřeních jsou zahrnuty ve zvláštní tabulce. Předpokládá se, že databáze se bude rozšiřovat v počtu dalších možných vyšetření a popřípadě v expanzi na jiná nemocniční oddělení. Pacient tak bude mít ve

správě pacientů uvedeny jen své osobní informace a jeho vyšetření budou provázána s dalšími tabulkami pomocí identifikátoru *Patient_ID*.

Tabulka 5: Uživatel

Staff						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Staff_ID	Integer	-	PK	Ne	Ano	ID uživatele
Name	Varchar	100	Ne	Ne	Ne	Jméno a příjmení uživatele
User_name	Varchar	50	Ne	Ne	Ano	Uživatelské přihlašovací jméno
Password	Varchar	50	Ne	Ne	Ne	Heslo
Job_position	Varchar	50	Ne	Ne	Ne	Pracovní zařazení
Note	Varchar	100	Ne	Ano	Ne	Poznámka

Tabulka Uživatel slouží výhradně pro zadání osobních informací o uživateli. V této tabulce nejsou uvedena jednotlivá práva daného uživatele. Na základě práv se budou lišit přístupy do konkrétních aplikací a z hlediska budoucího rozšiřování databáze bylo výhodné vytvořit tabulku s právy zvlášť. Administrátor tak bude mít přístup k vytváření nových práv v závislosti na tvorbě nových aplikací. Jelikož vazba mezi tabulkami *Uživatel* a *Role* je typu M:N (tedy jeden uživatel může mít více rolí, a zároveň jedna role může být přiřazena více uživatelům), bylo tedy nutné vytvořit také vazební tabulku Role uživatele, která prováže tyto dvě tabulky na základě jejich primárních klíčů.

Tabulka 6: Role uživatele

Staff_role						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Staff_role_ID	Integer	-	PK	Ne	Ano	ID
Staff_ID	Integer	-	FK	Ne	Ano	ID uživatele
Role_ID	Integer	-	FK	Ne	Ano	ID role

Tabulka 7: Role

Role						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Role_ID	Integer	-	PK	Ne	Ano	ID role
Name	Varchar	40	Ne	Ne	Ne	Název role
Role_key	Integer	-	Ne	Ne	Ne	Klíč role

Tabulka 8: Vyšetření

Examination						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Examination_ID	Integer	-	PK	Ne	Ano	ID vyšetření
Date_time	Timestamp	-	Ne	Ne	Ne	Datum vyšetření
Patient_ID	Integer	-	FK	Ne	Ano	ID pacienta
Examination_type	Integer	-	FK	Ne	Ano	Identifikátor druhu vyšetření
Note	Varchar	40	Ne	Ano	Ne	Poznámka pracovníka
Staff_ID	Smallint	-	FK	Ne	Ano	Uživatel, který prováděl vyšetření

Entita Vyšetření je založena tak, aby se zde mohly ukládat informace z různých vyšetření, z různých laboratoří či z různých oddělení v nemocnici. Uloží se zde datum a čas daného vyšetření, k tomuto vyšetření se přiřadí pacient, který ho vykonal, poté je zde uveden typ daného vyšetření, poznámka vložená pracovníkem a jméno pracovníka, který dané vyšetření s pacientem prováděl.

Tabulka 9: Typ vyšetření

Examination type						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Examination_type_ID	Integer	-	PK	Ne	Ano	ID typu vyšetření
Department_ID	Integer	-	FK	Ne	Ano	ID oddělení
Name	Varchar	50	Ne	Ne	Ne	Název typu vyšetření

Tabulka 10: Oddělení

Department						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Department_ID	Integer	-	PK	Ne	Ne	ID oddělení
Name	Varchar	50	Ne	Ne	Ne	Název oddělení
Note	Varchar	150	Ne	Ano	Ne	Popis

Tabulka Typ vyšetření a Oddělení opět poukazuje na nutnost oddělení těchto tabulek pro budoucí expanzi systému na vice oddělení. Při tvorbě této práce, bude v databázi uvedeno jen oční oddělení. Pokud se však systém rozšíří na další oddělení, nebude potřeba pro oddělení zakládat nové tabulky v databázi. Pouze se do existující tabulky zapíšou dané informace, přidělí se danému oddělení identifikátor a vytvoří se nový typ vyšetření. A pokud pacient toto vyšetření provede, všechny informace se promítnou do jeho „elektronické karty".

Tabulka 11: Parametry Hue testu

Hue_params						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Hue_params_ID	Integer	-	PK	Ne	Ano	ID Hue parametru
Examination_ID	Integer	-	FK	Ne	Ano	ID vyšetření
Test_type	Integer	-	Ne	Ne	Ne	Typy testu na barvocit
Manual	Integer	-	Ne	Ne	Ne	Manuální zadávání (Ano/ne)

Tabulka 12: Výsledky Hue testu

Hue_examination_results						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Hue_examination_results_ID	Integer	-	PK	Ne	Ano	ID výsledku
Examination_ID	Integer	-	FK	Ne	Ano	ID vyšetření
Color_index	Integer	-	Ne	Ne	Ne	Definice barvy čtverečku
Color_order	Integer	-	Ne	Ne	Ne	Pořadí seřazených čtverečků

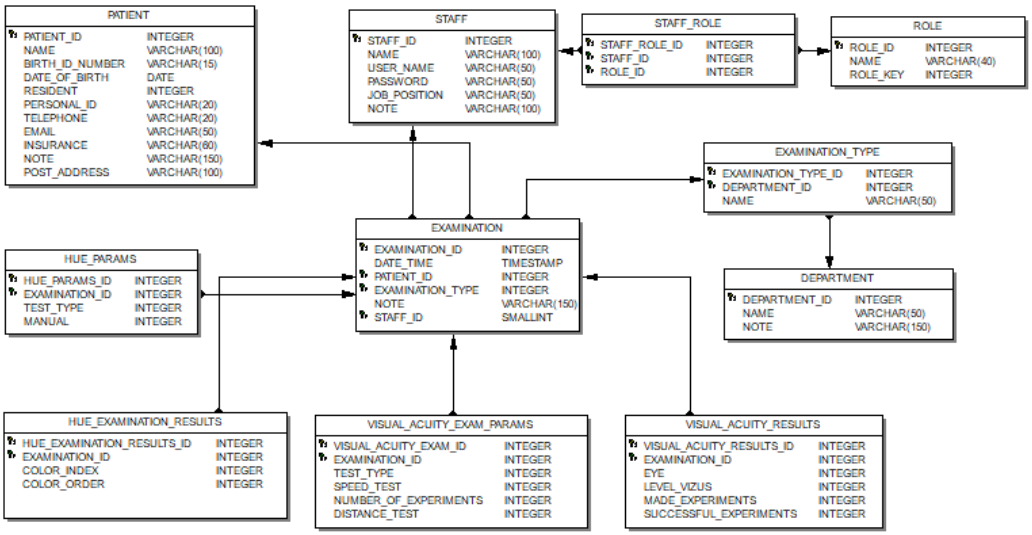
Tabulka 13: Parametry testu zrakové ostrosti

Visual_acuity_exam_params						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Visual_acuity_exam_params_ID	Integer	-	PK	Ne	Ano	ID parametru
Examination_ID	Integer	-	FK	Ne	Ano	ID vyšetření
Test_type	Integer	-	Ne	Ne	Ne	Typ testu
Speed_test	Integer	-	Ne	Ne	Ne	Rychlost podle počtu znaků
Number_of_experiments	Integer	-	Ne	Ne	Ne	Počet pokusů
Distance_test	Integer	-	Ne	Ne	Ne	Vzdálenost testu

Tabulka 14: Výsledky testu zrakové ostrosti

Visual_acuity_results						
Název	**Typ**	**Velikost**	**Klíč**	**Null**	**Index**	**Popis**
Visual_acuity_results_ID	Integer	-	PK	Ne	Ano	ID výsledku
Examination_ID	Integer	-	FK	Ne	Ano	ID vyšetření
Eye	Integer	-	Ne	Ne	Ne	Levé / pravé oko
Level_vizus	Integer	-	Ne	Ne	Ne	Úroveň viza pro každé oko
Made_experiments	Integer	-	Ne	Ne	Ne	Provedené pokusy
Successful_experiments	Integer	-	Ne	Ne	Ne	Úspěšné pokusy

V návrhu databáze jsou vytvořeny také tabulky pro další aplikace, které se budou na oční klinice využívat. Jedná se o aplikaci pro vyšetření barvocitu (HUE test), kterou vytvořila Ing. Monika Borová a aplikaci pro testování zrakové ostrosti, kterou vyvíjí Bc. Michaela Šidíková.



Obrázek 13: Datová analýza databáze

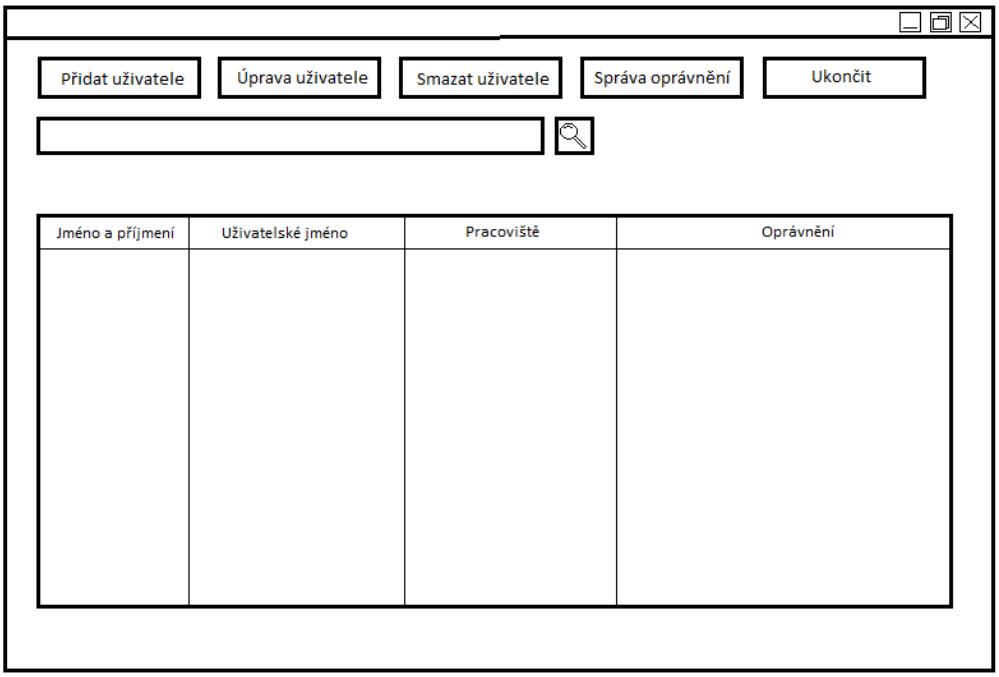
Z datové analýzy je patrné, že základním kamenem databáze je tabulka Examination. Ta obsahuje několik cizích klíčů. Do této entity se zapisují primární klíče z tabulek Pacient, Typ vyšetření a Uživatel. Na tyto tabulky je se přímo odkazuje dle jejich identifikátorů. ID vyšetření se jako cizí klíč uplatňuje v tabulkách, které slouží pro klientské aplikace. V parametrech a výsledcích těchto vyšetření je potřeba uvádět, který pacient jej vykonal, kdy, a jaký uživatel mu test spustil. Proto je to tabulka s nejvíce vazbami.

5.2 Modul administrace

Úkolem modulu administrace je možnost přidávat a upravovat uživatelské účty, včetně jejich uživatelských jmen a hesel. Požadavkem FNO bylo také vytvořit v administrátorské aplikaci práva, která se budou přiřazovat jednotlivým uživatelům, dle jejich funkcí. V databázi tak budou uvedeni pracovníci oční kliniky, kteří budou dané aplikace obsluhovat. Každé oprávnění bude obsahovat svůj číselný klíč, díky kterému bude administrator mít možnost určit, ke které aplikaci slouží jaký klíč. Uživatelé poté budou mít přiřazeny potřebné klíče. Při přihlášení se nejdříve ověří, jestli se daný číselný klíč, který mý uživatel přiřazen shoduje s daným číselným klíčem konkrétní aplikace. Byla navržena následující oprávnění do konkrétních aplikací:

- **Administrátor** – jedná se o základní roli. Administrátor bude mít přístup k modulu administrace, ve kterém bude přidávat nové uživatele a přiřazovat jim daná práva. Bude přidělovat uživatelům také jejich uživatelská jména a hesla.
- **Správa pacientů** – uživatel s tímto právem bude mít možnost přihlásit se do modulu pro správu pacientům. Bude mít tedy kompetence k přidávání nových pacientů a jejich úpravě. Bude mít možnost také nahlédnout jaká vyšetření a kdy pacient absolvoval. Toto právo budou mít přiřazeno především sestry na příjmu pacientů.
- **Testování barvocitu** – k tomuto vyšetření budou mít přístup zejména sestry v laboratoři, které budou vyšetření s pacientem provádět. Při otevření této aplikace si již zvolí vytvořeného pacienta, kterého přidala ve správě pacientů sestra na příjmu.
- **Testování zrakové ostrosti** – právo k tomuhle vyšetření bude mít také sestra v laboratoři, stejně jako při testování barvocitu.

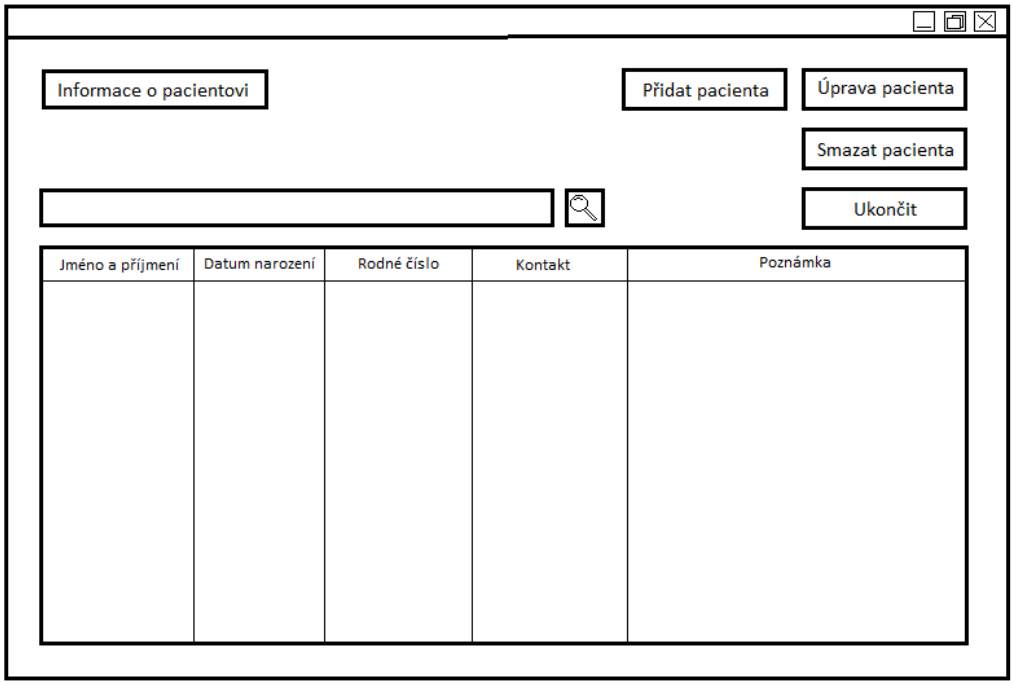
Bylo navrženo také grafické gui na základě potřebných funkcí modulů. Muselo tak obsahovat seznam uživatelů, seznam práv u každého uživatele, tlačítka pro přidání či úpravu uživatele a tlačítko pro otevření dalšího GUI správy oprávnění.



Obrázek 14: Grafické GUI modulu administrace

5.3 Modul pro správu pacientů

Úkolem modulu pro správu pacientů je možnost přidávat a upravovat údaje o pacientech. V tomto modulu se uživatelé také mohou podívat na proběhlá vyšetření pacienta, mohou si zobrazit parametry daných vyšetření, ale nemohou přes tuto aplikaci vyšetření provést ani se podívat na jeho výsledky. Bylo také jako v předchozím modulu vytvořeno grafické gui, kde byly všechny tyto funkčnosti zohledněny. Hlavní formulář musel tedy obsahovat seznam pacientů, tlačítka pro přidání a úpravu pacienta, tlačítko, které nám otevře další gui pro zobrazení informací o daném pacientovi. Každý návrh musí také obsahovat tlačítko pro ukončení aplikace.



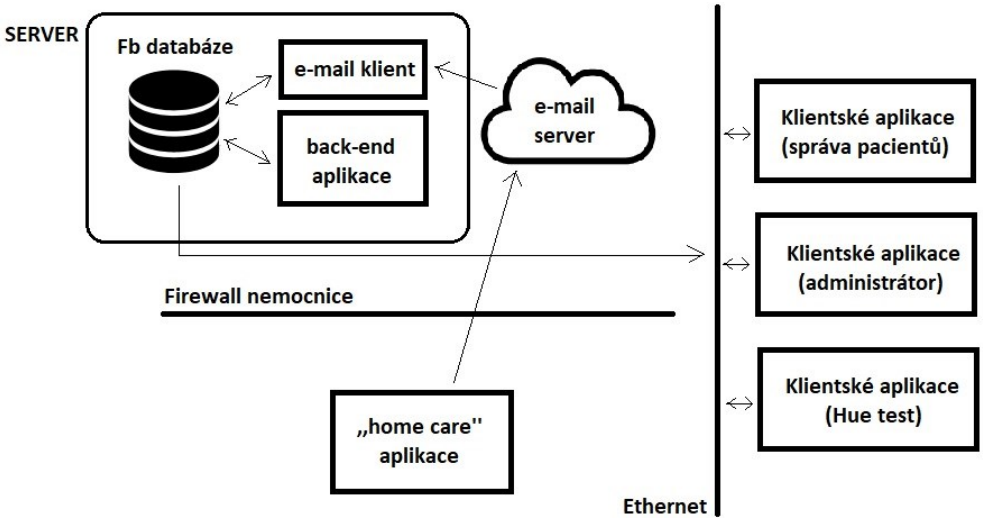
Obrázek 15: Grafické GUI modulu správy pacientů

6 Implementace modulů administrace a správy pacientů v jazyce C#

Celkový informační systém a jeho moduly byly vyvíjeny v Microsoft Visual Studiu pro platformu Microsoft .NET Framework 4.5 a je dostupný v operačních systémech Windows 7, 8, 10 a Windows Vista.

Databáze, vytvořená v této diplomové práci, tvoří jádro informačního systému a běží na serveru Fakultní nemocnice v Ostravě. K této databázi je možné připojit aplikace pro oční kliniku, které mohou fungovat jak na PC, tak na mobilním telefonu či tabletu. Aplikace, které jsou nainstalovány na oční klinice spolu komunikují prostřednictvím back-end aplikace, která běží na serveru. Aplikace mohou být také typu ,,home care'', které si může pacient spustit doma sám. Data získaná z daného vyšetření může odeslat do nemocničního zařízení. K tomu slouží e-mail server. Na serveru běží také e-mail klient, který získaná data ověří a uloží do databáze.

Tato diplomová práce je součástí širšího projektu pro Oční kliniku, na kterém se podílí další studenti. Dále v textu se budeme zabývat pouze implementací aplikací vytvořených v rámci této práce.



Obrázek 16: Architektura informačního systému

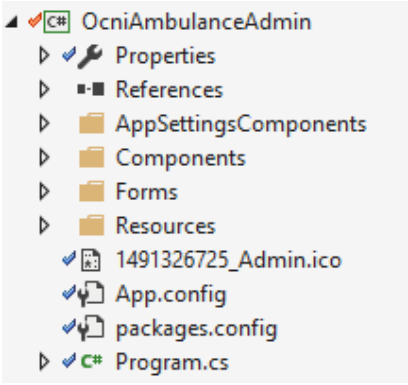
6.1 Struktura aplikace

Celá aplikace tvořená v prostředí Visual Studia sestává z několika modulů, které se vyvíjely v rámci jednoho projektu s názvem *OcniAmbulance*. Vyvíjel se zde modul pro správu pacientů *(OcniAmbulanceUser),* administrátorský modul *(OcniAmbulanceAdmin)* a také aplikace pro testování barvocitu *(HueTest)*.

Byla zde vytvořena také složka *Shared*, ve které byly vytvářeny formuláře a metody, které se použily v modulech vicekrát. Objevují se zde formuláře, které mohou být znovupoužitelné v dalších aplikacích. Jsou zde vytvořeny UserControly, což je grafické gui, které sestavíme z grafických komponent a mohou se poté vícenásobně vkládat do jednotlivých dialogů. Tato složka tedy ušetří čas i práci při psaní redundantního kódu.

V této složce je umístěno celé připojení do databáze, které budou využívat všechny aplikace a mohou se na něj odkazovat. Všechny složky ve sdíleném projektu Shared jsou popsány v následující podkapitole Implementace sdílených částí.

V každém projektu jsou vytvořené složky pro lepší přehlednost. Jsou zde složky Components, které obsahují třídy, složky Forms, které obsahují jednotlivé formuláře, složky Resources, ve kterých jsou seznamy textových řetězců a SQL dotazů a další složky, dle potřeby každé aplikace. Na následujícím obrázku je uveden projekt *OcniAmbulanceAdmin* a jeho struktura.



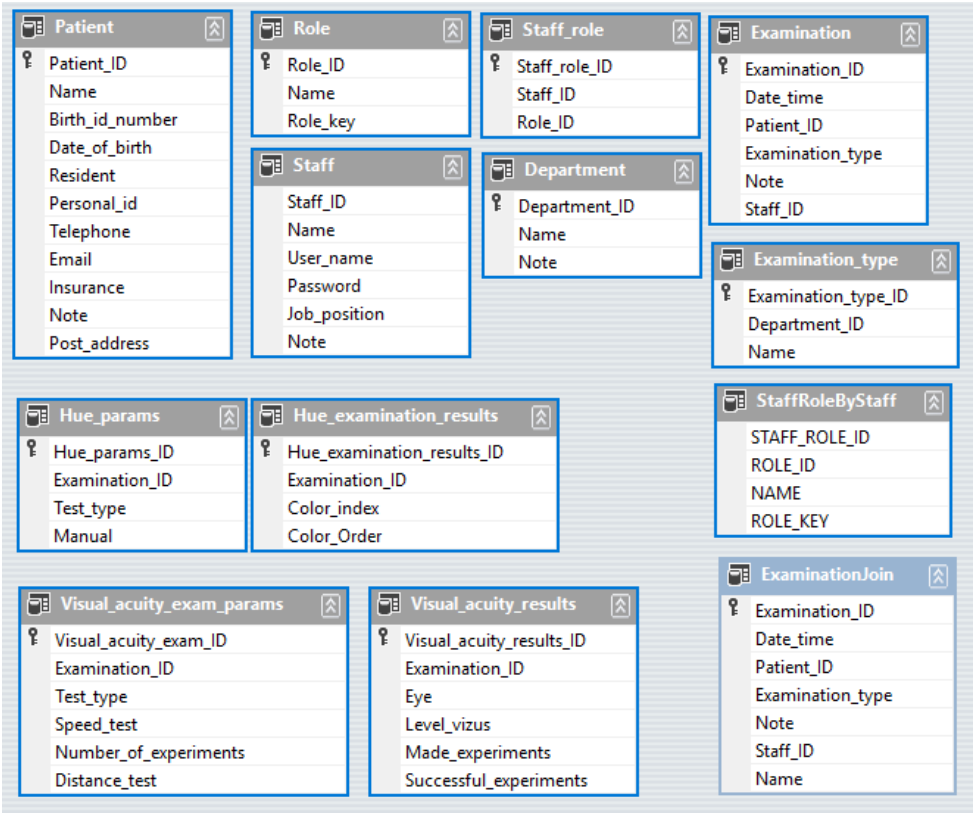
Obrázek 17: Struktura administrátorské aplikace

6.2 Implementace sdílených částí

Sdílené části v této práci zahrnují funkcionality a dialogy, které budou využívat vývojáři aplikací, kteří se budou připojovat do jednotné databáze vytvořené v této diplomové práci. Mohou zde využít přihlašovací formulář a také User Controly pro implementaci seznamu pacientů či zobrazení jejich vyšetření.

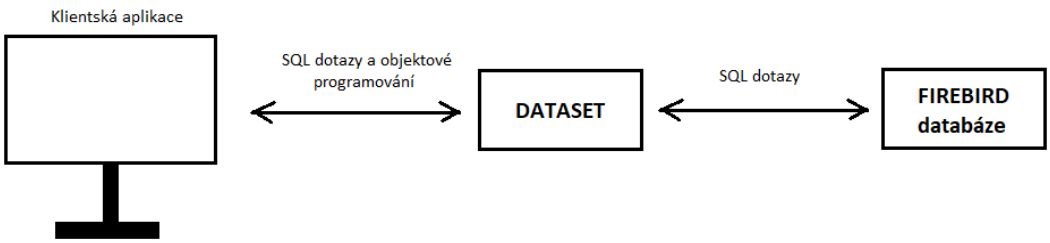
6.2.1 Vytvoření DataSetu

Ve sdílené složce Shared byl vytvořen Dataset, který si můžeme představit jako obraz databáze. Vytvořily se zde tabulky s příslušnými relacemi a jsou do nich uchovávány informace během procesu. Po dokončení všech operací se zpřístupní originální Firebird databáze a všechny změny jsou do ní zapsány. Tabulky zde vytvořené se nazývají *DataTable*, jejich sloupce *DataColumn* a řádky *DataRow*.



Obrázek 18: Vytvořený DataSet ve Visual Studiu

DataSet využíváme z důvodu rozdílného přístupu v objektově orientovaném programování a v relačních databázích. Relační databáze nevyužívají objektově orientovaný přístup a tak bylo nutné vytvořit tento DataSet, který nám umožňuje pracovat s databází jako s objektem. Komunikace však probíhá stále v jazyce SQL.



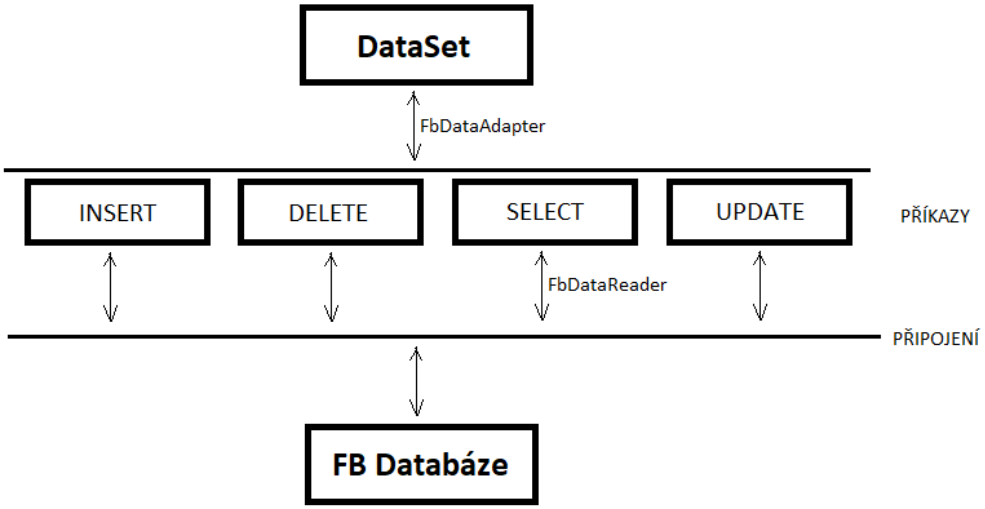
Obrázek 19: Propojení objektového programování a Firebird databáze

DataSet komunikuje s databází pomocí instance tříd rozhraní FbDataAdapteru. Třída DbTools obsahuje metody, díky kterým jsou do databáze vysílány příkazy jako např. select, insert, delete atd. Pro čtení z databáze se jedná o metodu SelectQuery:

```
public static void SelectQuery(FbConnection connection, DataTable table,
string query)
```

A pro zápis do databáze slouží metoda ExecuteQuery:

```
public static void ExecuteQuery(FbConnection connection, string query)
```



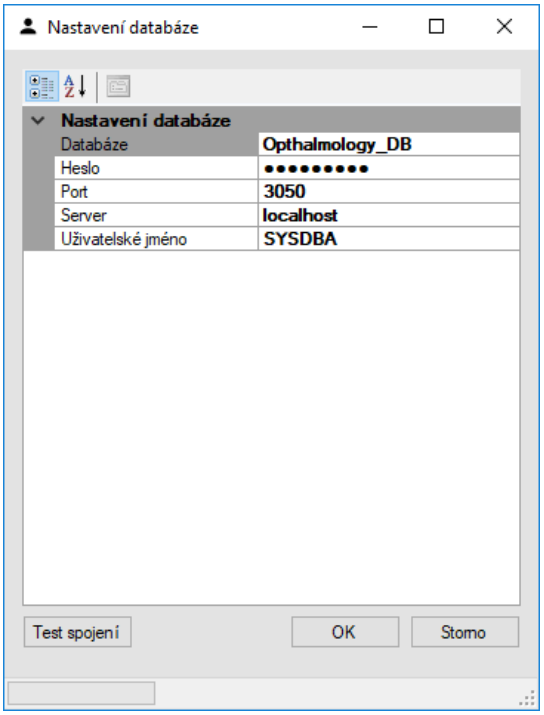
Obrázek 20: Princip komunikace mezi DataSetem a databází

6.2.2 Připojení k databázi

Ve sdílené složce je umístěno celé připojení do databáze. Obsahuje třídu *DbSettings,* ve které jsou definovány údaje, které je třeba vyplnit pro připojení do databáze. Třída obsahuje jméno databáze *Database*, uživatelské jméno *User_Name*, heslo *Password*, číslo portu *Port* a název serveru *Server_name*. Tyto údaje vyplňuje administrator při nastavování databáze. Další třída je *DbTools*, která pomocí *ConnectionString* připojuje jednotlivé moduly k databázi. Díky tomu můžeme využívat SQL dotazy a hledat či vkládat údaje do databáze.

```
string connectionString =
                "User=SYSDBA;" +
                "Password={0};" +
                @"Database={1};" +
                "DataSource={2};" +
                "Port={3};" +
                "Dialect=3;" +
                "Charset=NONE;" +
                "Role=;" +
                "Connection lifetime=15;" +
                "Pooling=true;" +
                "MinPoolSize=0;" +
                "MaxPoolSize=50;" +
                "Packet Size=8192;" +
                "ServerType=0";
            connectionString = string.Format(connectionString,
                dbParameters.Password,
                dbParameters.Database,
                dbParameters.Server,
                 dbParameters.Port);
```

V tomto formuláři je zapotřebí nastavit všechny potřebné informace pro připojení do dané databáze. Pro zjednodušení zadávání cesty k souboru je vytvořen tzv. *alias*, což je název, pod kterým se skrývá přístupová cesta k databázi v počítači. Tento alias se vytváří v souboru *aliases.conf*, který se nachází v instalační složce Firebirdu. Tento alias poté zádáváme ve formuláři místo samotné cesty k souboru. Při vyplnění všech údajů si ověříme spojení pomocí tlačítka *Test spojení* ve spodní části formuláře. Pokud jsou všechna nastavení správně, vypíše se hláška „Spojení s databází je v pořádku." Můžeme tedy stisknout *Ok* a pokračovat v přihlášení do aplikace. Toto připojení stačí nastavit pouze jednou při samotném zavádění aplikace a nastaví jej administrátor, který zná potřebné údaje ke spuštění tohoto nastavení. Poté se již uživatel o přístup k databázi nemusí starat.



Obrázek 21: Formulář pro nastavení databáze

6.2.3 Přihlášení do aplikací

Přihlašovací formulář *LoginForm* je zapotřebí v každé aplikaci. Tento dialog je tedy uveden ve sdílené složce a je vytvořen z nástrojů dostupných ve Visual Studiu. Jedná se o *textboxy, labely* a *tlačítka*. Uživatel vyplní své údaje a stiskne tlačítko Přihlásit se. Správné přihlášení uživatele se ověří metodou *CheckCredentials.* Tato metoda je založena na porovnání informací zadaných uživatelem a informacemi uloženými v databázi. Systém pomocí SQL dotazu *SELECT* vybere z databáze uživatele, jehož údaje se shodují s údaji zadanými uživatelem. Pokud nastane shoda, systém si zapamatuje ID daného uživatele a pokračuje se k ověřování oprávnění. Oprávnění uživateli přiřazuje pouze administrator. Ověřování probíhá také pomocí SQL příkazu *SELECT_STAFF_ROLE*, který vrátí ID oprávnění, která byla danému uživateli přiřazena. SQL příkaz, který vybere ID role daného uživatele vypadá takto:

```
select sr.Staff_role_ID, r.Role_ID, r.Name, r.Role_key
from Staff_role as sr
inner join Role as r on r.Role_ID = sr.Role_ID
where sr.Staff_ID={0};
```

Za zástupný znak {0} se v kódu vyplní `user.Staff_ID`, což je ID uživatele, který se přihlásil. Pokud se oprávnění uživatele shoduje s klíčem role dané aplikace, uživateli se otevře hlavní strana zvolené aplikace. Do vytvořené proměnné *LoggedUser* se uloží ID uživatele, který se do dané aplikace přihlásil. Díky této proměnné se nám ke každému provedenému vyšetření uloží jméno uživatele, který jej prováděl. Tento přihlašovací formulář je použitelný v každé aplikaci a můžeme ho zavolat ve třídě Program.cs každé aplikace. V této třídě použijeme *using*, který nám umožní použít třídy jinných

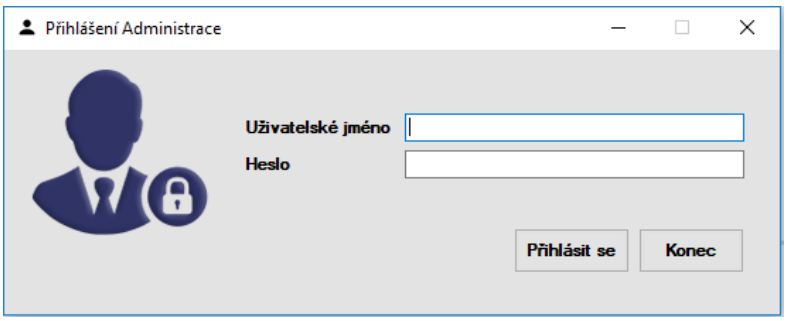
jmenných prostorů. V tomto případě zavolání přihlašovacího formuláře i s jeho funkčnostmi vypadá takto:

```
using Shared.Forms;

LoginForm login = new LoginForm();
login.SetCaption(Language.APP_NAME);
login.RoleKeys.Add(2);
login.UserName = Properties.Settings.Default.LastUserName;
login.SetConnection(Program.DatabaseConnection);
login.ShowDialog();
Properties.Settings.Default.LastUserName = login.UserName;
Properties.Settings.Default.Save();
```

Všechny zmíněné metody jsou nadefinovány ve sdílené složce Shared a jsou definovány jako veřejné. Metoda *SetCaption* vypíše název právě otevírané aplikace do hlavičky přihlašovacího formuláře. Je zde využito souboru *Language.resx*, který nám úvádí výčet textových řetězců. Pod proměnnou *APP_NAME* se tedy skrývá konkrétní název aplikace. Funkce *RoleKeys.Add* nám uvádí klíč role, který je potřebný pro přístup k aplikaci. Uživatel, který se chce do aplikace přihlásit musí mít oprávnění s daným klíčem (v tomto případě 2). Pokud má jiný klíč role, nemá přístup spustit danou aplikaci. Následující řádek *login.UserName* slouží k uložení naposledy přihlášeného uživatele a pomocí metody *get{}set{}* zůstane v přihlašovacím formuláři dané aplikace vyplněno uživatelské jméno. Uživatel pracující opakovaně s danou aplikací tak bude zadávat pouze své heslo. Metoda *SetConnection* se připojuje k databázi pomocí FB klienta.

Pokud uživatel nezadá své údaje správně, aplikace vyvolá vyjímku a vypíše pomocí metody *MessageBox.Show* hlášku: „Špatné uživatelské jméno nebo heslo.'' Pokud zadá údaje správně, ale nemá právo otevírat danou aplikaci, vypíše se hláška: „Nemáte právo se přihlásit do této aplikace.''



Obrázek 22: Přihlašovací formulář

6.2.4 User Controly

Ve sdílené částí aplikace *Shared* byly naimplementovány také ovládací prvky, které se budou využívat ve více aplikacích. Vyhneme se tak vytváření několika stejných formulářů, které mají stejné vlastnosti. Jedná se o ovládací prvek, který je do aplikace vložen pomocí funkcí *Add* → *New Item* a v seznamu vybereme položku User Control. Tento prvek si naformátujeme pomocí ovládacích prvků z *Toolboxu* dle vlastních potřeb. Můžeme zde vytvořit také potřebné metody a funkce tlačítek. Poté se tento User Control uloží do prvků Toolboxu a můžeme s ním zacházet, jako s ostatními prvky, je tedy znovupoužitelný. V potřebných formulářích si tedy jen vybereme daný User Control z panelu Toolbox.

Obrázek 23: User Control pro vyhledávání pacienta ze seznamu

Byl vytvořen User Control s názvem *UcPatientList,* který slouží pro načtení pacientů z databáze do připravené tabulky. Načtení probíhá pomocí metody SelectQuery, uvedené ve třídě *DbTools,* a pomocí SQL dotazu *SELECT_ALL_PATIENTS*. Je zde dlouhý *TextBox*, který umožňuje vyhledat si pacienta v databázi podle jména či rodného čísla. Toto vyhledávání zajišťuje událost *TextChanged* a SQL dotaz *Search_Patient*.

Formulář obsahuje také seznam pacientů. Tento seznam je reprezentován prvkem *dataGridView* a v tomto seznamu se nám zobrazují všichni pacienti v databázi. Je zde využito třídy *BindingSource*, která zajišťuje vazbu mezi ovládacím prvkem DataGridView a zdrojem dat. Vytvoří také interface seznamu a zjednoduší nám práci při jeho tvorbě. *DataGridView* je nastaveno tak, abychom do něj nemohli nic vpisovat, ani uprovavat informace. Slouží jen pro označení požadovaného pacienta. Systém uloží právě vybraného pacienta do proměnné *SelectedRow* a s touto proměnnou můžeme dále pracovat a zobrazovat si například údaje pacienta či jeho vyšetření.

Obrázek 24: User Control pro zobrazení informací o pacientovi

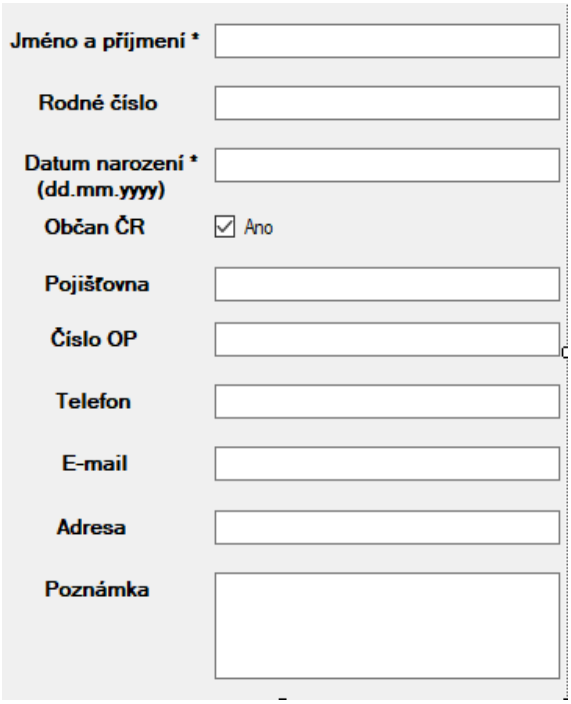
Následující User Control s názvem *UcPatientInfo*, slouží pro načtení dat konkrétního pacienta vybraného ze seznamu. Do proměnné *SelectedRow* zmíněné výše, se uložilo ID pacienta, na kterého bylo kliknuto v seznamu pacientů. Pomocí metody *SetControl* byly přiřazeny osobní informace o pacientovi do příslušných TextBoxů tohoto UserControlu. V seznamu se zobrazí jenotlivá vyšetření pacienta a objeví se parametry daného vyšetření. Tyto informace již nejde upravovat. Seznam vyšetření je vytvořen pomocí *DataGridView* a je v něm uveden název daného vyšetření, jeho datum a jméno uživatele, který s pacientem vyšetření prováděl. Datum návštěvy se vygeneruje během daného vyšetření pomocí funkce:

```
DateTime.Now.ToUniversalTime()
```

Tato funkce uloží do databáze datum a konkrétní čas. Typ vyšetření se přenese z dané aplikace. Ve třídě Program.cs každé aplikace je uvedeno ID typu vyšetření. To je uloženo do proměnné

```
public const int EXAMINATION_TYPE_ID = 1
```

Při ukládání vyšetření do databáze, se pomocí SQL příkazu tedy uloží i toto ID, které se poté v UserControlu pro přehled vyšetření znovu zavolá a místo ID se zde vypíše název daného vyšetření. Ve spodní části obrazovky je *ListView*, které vypíše údaje o daném vyšetření. Každá aplikace má svou sadu parametrů a ty mohou nabývat různých hodnot. Tato funkčnost je implementována pomocí techniky **Dependency Injection,** neboli **Vkládání závislostí**. Díky této technice můžeme využívat komponenty, které vytvořil tzv. *Poskytovatel závislostí* a můžeme je využívat v našem objektu. Jednotlivé parametry a jejich hodnoty se totiž budou měnit v závislosti na typu vyšetření. Tato funkce byla implementována pomocí třídy *PatientInfoParameters*. Každá aplikace má zvláštní třídu pro tyto parametry. Pomocí SQL dotazu se vytáhne z databáze název parametru a jeho hodnota. SQL dotaz je implementován v metodě *GetExaminationParameters* a obsahuje také podmínku *if*, která ověří jestli existují v databázi parametry vybraného vyšetření, a pokud ano, vrací kolekci daných parametrů a uloží název parametru do kolekce *Name* a hodnotu parametru do kolekce *Value*. Takových parametrů může být u daného vyšetření vice a ty se poté zobrazí v ListView uvedeného UserControlu.



Obrázek 25: UserControl pro uložení nového pacienta

Posledním implementovaným UserControlem, je prvek pro zadávání či úpravu pacienta. Tento UserControl se využívá pouze v aplikaci pro správu pacientů. Je zde však použit jak ve formuláři pro vytvoření nového pacienta, tak ve formuláři pro úpravu pacienta. Proto byl vytvořen jako znovupoužitelný UserControl. Tento prvek je sestaven z *Textboxů* a *Labelů*. Je zde jeden *CheckBox* pro zatrhnutí občanství daného pacienta. Pomocí události *Resident.Checked* systém hlídá, zda je políčko zatrhnuto nebo ne. Daný stav se poté zapíše jako logická 1 nebo 0 a uloží se takto do databáze.

6.3 Implementace modulu pro správu pacientů

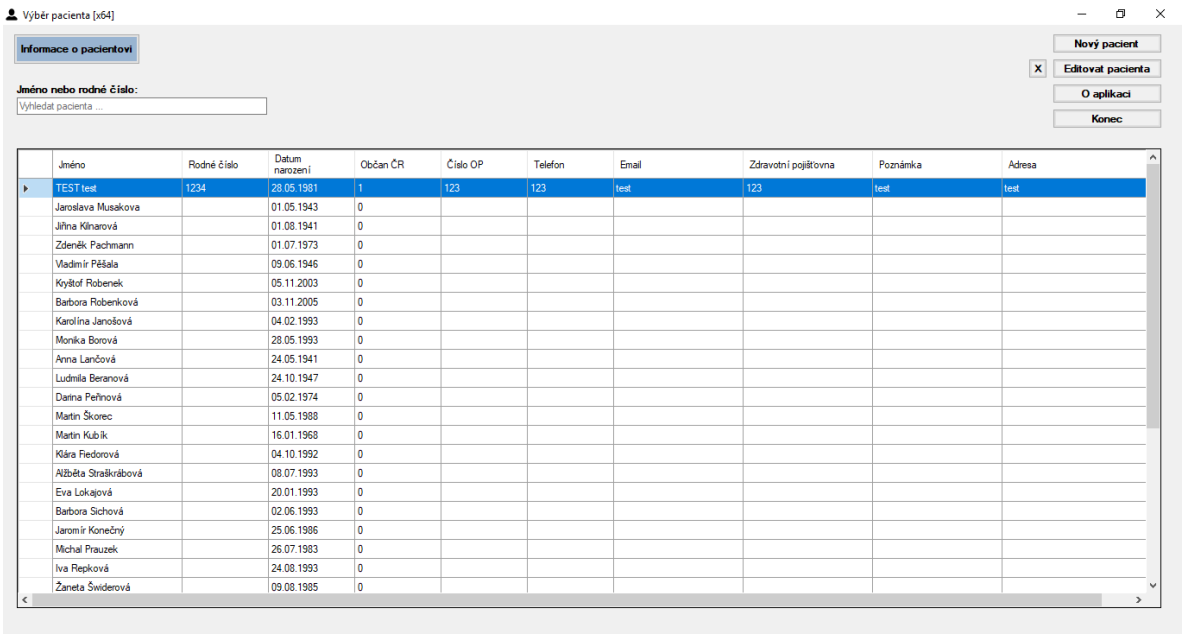
Tento modul slouží primárně k vyhledání a zadávání nových pacientů. Přístup do této aplikace mají všichni uživatelé, kteří vlastní danou roli s klíčem číslo 2. Přihlášením do této aplikace se uživateli zobrazí hlavní formulář, ve kterém jsou použity základní ovládací prvky *Toolboxu*. V tomto formuláři bylo využito předpřipraveného *User Controlu* z panelu *Toolbox* s názvem *UcPatientList*. Aby se nám aplikace propojila s UserControly implementovanými ve složce Shared, je zapotřebí vytvořit referenci na tuto složku pomocí příkazu

```
using System.Windows.Forms.
```

U každého pacienta můžeme vidět základní informace o něm. Pro uživatele je však skryto ID pacienta. Veškeré hledání v databázi se v C# realizuje SQL dotazy. Používaný SQL dotaz pro vyhledání pacienta vypadá např. takto:

*select * from patient where Patient_Id={0}.*

Tento SQL dotaz nám vrátí všechny údaje o jednom pacientovi. Využíváme tady zástupného znaku {0}, místo kterého se nám v tomto případě doplní ID pacienta, vybraného například ze seznamu pacientů.



Obrázek 26: Hlavní strana aplikace pro správu pacientů

V hlavním formuláři jsou místo UserControlu, který jsme si popsali výše naimplementovány také dvě tlačítka *Nový pacient* a *Editovat pacienta*. Tyto formuláře jsou typově stejné, pouze jinak naprogramované. Jedná se také o předdefinované UserControly s názvem *UcPatientData*. Jsou zde dvě povinná pole a těmi je jméno a datum narození pacienta. Pokud budeme chtít editovat pacienta vybraného ze seznamu, zobrazí se nám předvyplněný formulář s údaji, které jsme zadali při jeho vytváření. Tento přenos dat je prováděn pomocí ID pacienta, které je jedinečné. Kliknutím na daný řádek v seznamu pacientů se ID tohoto pacienta zapíše do veřejné proměnné *SelectedRow*. Při otevírání formuláře pro jeho úpravu si pomocí metod *SetPatientRow* a *SetControl* vytáhneme z databáze ID pacienta a jeho údaje přiřadíme do správných TextBoxů. Na hlavní obrazovce je také křížek, kterým můžeme pacienta vymazat. Systém je však vytvořen tak, abychom nemohli smazat pacienta, který již prováděl nějaké vyšetření, a to je uloženo v databázi. To ošetříme pomocí několika vyjímek. Nejprve zjistíme, pomocí příkazu *if*, jestli byl vybrán nějaký pacient. Pokud ano, pomocí SQL příkazů se ověří, jestli nemá pacient přiřazeno nějaké z vyšetření. Při takovém pokusu systém vypíše chybovou hlášku: „*Pacient má již provedeno vyšetření a nelze jej proto smazat.''*.

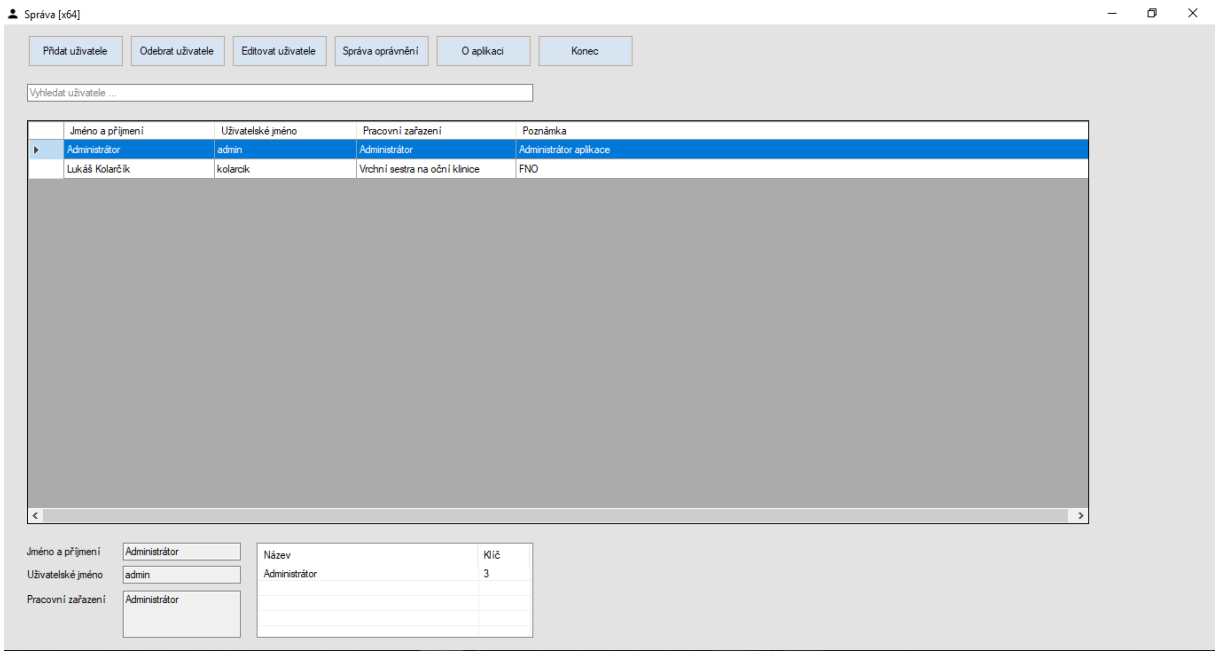
Stisknutím tlačítka *Informace o pacientovi* otevřeme další formulář, který se týká jednoho konkrétního pacienta, kterého jsme vybrali v seznamu. V tomto formuláři byl opět využit User Control s názvem UcPatientInfo se všemi jeho parametry a vlastnostmi, které již byly popsány výše. Veškeré data se zapisují a získávají z databáze. K ní přistupujeme pomocí SQL dotazů. Tyto dotazy ukládáme do souboru pro ukládání zdrojů s názvem *Resource File*. V takovém souboru s příponou .resx jsou ukládány také textové řetězce. Usnadní nám to práci s těmito řetězci a zjednoduší to práci při změně daného textu. Řetězci přiřadíme název a tím se budeme v kódu odkazovat na daný řetězec.

Na následujícím obrázku je uvedena ikona celé aplikace, tak jak ji uvidí uživatel na ploše PC. Aplikace obsahuje také ikony jednotlivých formulářů.

Obrázek 27: Ikona aplikace pro správu pacientů

6.4 Implementace modulu administrace

Tento modul je dostupný pouze uživateli, který má oprávnění administrátora. V této aplikaci má přístup k seznamu uživatelů. Uživateli mohou být jak zdravotní sestry, tak lékaři, či biomedicínští technici. Otevřením aplikace se zobrazí hlavní formulář, v jehož středu je seznam, vytvořený pomocí *DataGridView,* který uvádí základní informace o uživatelích. Je zde skryto, jak *ID uživatele*, tak jeho heslo k přihlašování. Ve spodní části obrazovky jsou *TextBoxy*, ve kterých se vypíší informace o uživateli vybraného ze seznamu a je zde ListView, ve kterém se zobrazí oprávnění, které daný uživatel má.



Obrázek 28: Hlavní strana administrátorské aplikace

Možností administrátora je vytváření nových uživatelů. Vytvoří jej tak, že klikne na tlačítko Přidat uživatele. V novém formuláři se zobrazí prázdné *TextBoxy*, do kterých administrátor vypíše potřebné údaje. Stanovuje také uživatelské jméno a heslo, kterým se uživatel bude přihlašovat do potřebné aplikace. Heslo je zde skryto a zašifrováno algoritmem MD5, který ze vstupu vytvoří jedinečný klíč pevně dané délky. V pravé části formuláře je *ListView*, ve kterém se nám zobrazí seznam dostupných oprávnění. Všechna uložená oprávnění se v seznamu načítají pomocí SQL příkazů. Administrátor zatrhne ta práva, která budou přidělena danému uživateli. Na hlavní obrazovce je také tlačítko *Editovat uživatele*, které nám otevře stejný formulář s vyplněnými údaji, které můžeme měnit. Otevření tohoto formuláře je svázáno s ID uživatele na kterého bylo kliknuto v hlavním seznamu.

Systém si zapamatuje dané ID a na základě něho poté v novém formulář vypíše data z databáze. Mazat uživatele může administrátor pouze v případě, že se daný úživatel nepodílel na žádném vyšetření a že nemá přiřazena žádná oprávnění. Ta můžeme uživateli odebrat ve formuláři pro úpravu uživatele.

Další položkou a možností administrátora je *Správa oprávnění*. V tomto formuláři vidíme všechna oprávnění vytvořená administrátorem a jejich klíče. Formulář pro přidání nové role se zobrazí kliknutím na dané tlačítko. Je potřeba jen vyplnit název a klíč role. Stejně můžeme také upravovat či mazat role pomocí tlačítek. Vymazat roli však jde pouze v případě, že není přiřazena žádnému uživateli.

Na následujícím obrázku je uvedena ikona celé aplikace.

Obrázek 29: Ikona administrátorské aplikace

6.5 Zpracování vyjímek

Jednotlivé funkce programu mohou za jistých okolností vyvolat vyjímečný chybový stav, který je zachycen systémem vyjímek. Ty jsou objektově orientované a jsou odvozeny z třídy `System.Exception`. Každá vyjímka obsahuje informace o důvodu vzniku. V programu je potřeba vyjímky zachytit a zpracovat. V této práci je pro zachycení vyjímek používán blok `try {}, catch {}`. V bloku `try` je naimplemntovaná samotná funkce metody a pokud vznikne vyjímka, je zachycena v bloku `catch` a vypsána. Za klíčovým slovem `catch` je uveden typ vyjímky, který může nastat. V následujícím příkladu je uveden typ `System.Exception`, který je společný pro všechny vyjímky a tak je zachytí všechny na jednom místě. Obecně vypadá blok pro zpracování vyjímek takto:
[9]

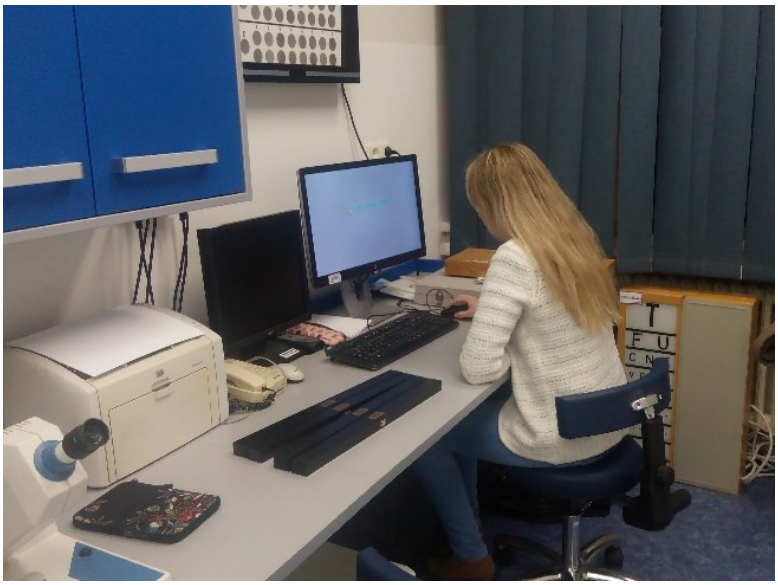
```
try
{
 //chraneny blok
}
catch(System.Exception ex)
{
 //spolecny obsluzny blok pro
 //vsechny vyvolane vyjimky
}
```

7 Testování funkčnosti modulů na oční klinice

Testování funkčnosti obou modulů probíhalo sestavením testovacího plánu. V tomto plánu jsou zahrnuty popisy jednotlivých funkčností a jejich očekávaný výsledek. Pokud byl zaznamenán neočekávaný výsledek a byla nalezena chyba, ihned se opravila a testovalo se znovu. Chyby nalezené při závěrečném testování ukázaly na chybné ošetření vyjímek a uživatelských vstupů. Testování probíhalo nejdříve na testovací databázi, kde již byli vloženi pacienti. Při testování ve vyplněné databázi bylo vše v pořádku.

Chybné ošetření vyjímek bylo zjištěno při testování s prázdnou databází. Na základě těchto chyb byl v systému vytvořen tzv. **`ErrorLog`**, což je zpráva o vzniklé chybě. Je naimplementován v každé aplikaci. Uvádí se v něm jméno dané aplikace a zaloguje chybovou hlášku do XML souboru pomocí třídy `StreamWriter`. Zde se vygeneruje datum a čas vzniklé chyby, verze systému, jméno uživatele a vygenerovaná chybová hláška. Tento `ErrorLog` poté usnadní správci hledání vzniklé chyby.

Tyto chyby se ošetřily a celý systém již byl v pořádku a oba moduly byly nainstalovány do laboratoře na oční kliniku. Po nainstalování byl poučen personál a byly vyzkoušeny opět všechny funkčnosti. Systém byl na kliniku nainstalován v listopadu 2017.



Obrázek 30: Pacienti, kteří byli vloženi do databáze a prováděli vyšetření barvocitu

Testování každého modulu začínalo testováním přihlašování. Bylo důležité, aby každá vzniklá chyba vypsala správnou hlášku. Uživatel tak pochopí, kde udělal chybu a zopakuje přihlášení. Mohou zde vzniknout situace, kdy uživatel zadá špatné přihlašovací údaje, nemá přístup k dané aplikaci nebo vznikne chyba při připojení do databáze. Všechny tyhle situace jsou ošetřeny vyjímkami pomocí bloků `try/catch`.

Po přihlášení na hlavní stranu aplikace pro správu pacientů je zde několik tlačítek, které musí fungovat správně. Bylo potřeba důkladně otestovat vyhledávání pacienta v seznamu. Při vyplňování textboxu pro vyhledávání se po každém písmenu nebo číslu aktualizuje seznam a vypíší se jen pacienti, kteří mají ve svém jméně uvedeno konkrétní písmeno nebo v rodném čísle dané číslo. Při zadávání nového pacienta byla stanovena povinná pole, která musí být vyplněna. Pokud vyplněna nejsou, systém

vyhodí adekvátní hlášku, která uživateli řekne, kde udělal chybu. Vyjímky bylo třeba ošetřit také při mazání pacienta. Nemůže být totiž smazán pacient, který již byl na klinice vyšetřen.

Ve formuláři, kde jsou uvedeny informace o pacientovi bylo nutné otestovat správné vyplnění údajů o vyšetřeních, které pacient provedl. Bylo nutné zde tedy otestovat techniku `Dependency Injection`, která je popsána v předchozí kapitole. Jedná se o správné zobrazení daných parametrů vyšetření. V tabulce vyšetření se objeví vykonané testy z různých klientských aplikací. Pokud se vyznačí řádek s příslušným vyšetřením, systém si zapamatuje typ daného vyšetření a přiřadí k tomuto vyšetření vhodné parametry a jejich hodnoty při konkrétním testu. Tento test se prováděl tak, že pacient provedl v aplikaci pro testování barvocitu vyšetření a poté se v modulu pro správu pacientů otevřely jeho informace a zde by mělo být dané vyšetření hned zapsáno včetně data, času a jména uživatele, který s pacientem test prováděl. Vše v tomto modulu probíhalo správně a podle protokolu.

Tabulka 15: Testovací protokol modulu pro správu pacientů

Modul správy pacientů	
Popis	**Očekávaný výsledek**
PŘIHLÁŠENÍ	
Správné přihlášení	Spuštení aplikace
Špatné přihlášení	Chybová hláška
HLAVNÍ STRANA	
Vyhledání pacienta	Filtrace seznamu dle jména nebo RČ
NOVÝ PACIENT	
Tlačítko nový pacient	Otevře se formulář pro zadání údajů
Uložení nového pacienta	Pacient se uloží do databáze, formulář se zavře a původní seznam se aktualizuje
Nevyplním povinná pole	Chybová hláška
Tlačítko zavřít	Formulář se zavře a zůstane zobrazena hlavní obrazovka se seznamem pacientů
EDITOVAT PACIENTA	
Tlačítko editovat vybraného pacienta	Otevře se formulář s vyplněnými údaji
Uložení změn	Tlačítkem uložit vložím nově změněné údaje do databáze. Formulář se zavře a seznam se aktualizuje.
Při editaci pacienta vymažu povinné pole	Chybová hláška
ODSTRANĚNÍ PACIENTA	
Tlačítko smazat pacienta	Pacient je smazán z databáze a seznam pacientů se aktualizuje
Mazání pacienta, který již byl vyšetřen	Chybová hláška
O APLIKACI	
Otevření údajů o aplikaci	Otevře se okno s informacemi o aplikaci
ÚDAJE O PACIENTOVI	
Tlačítko údaje o pacientovi	Otevře se okno s jeho vyplněnými údaji a vyšetřeními
Zobrazení podrobných údajů o vyšetření	Po kliknutí na dané vyšetření se zobrazí jeho parametry
Údaje o vyšetřeních	U vyšetření se zobrazí jeho datum a uživatel, který vyšetření provedl

KONEC APLIKACE	
Tlačítko konec	Stisknutím tlačítka konec se celá aplikace ukončí

Testování přihlášení v administrátorském modulu probíhalo stejně jako v modulu pro správu pacientů. Po správném přihlášení bylo nutné opět otestovat funkčnost seznamu uživatelů a veškerých tlačítek. V této aplikaci bylo náročnější propojení s databází, jelikož u každého uživatele se nevyplňovaly jen jeho osobní informace jako u pacienta, ale museli se mu také přiřadit oprávnění k jednotlivým aplikacím. Toto přiřazení funguje na základě vytvoření složitějších a rozsáhlých SQL dotazů přes více tabulek. Přiřazování práv bylo otestováno ve formuláři pro přidání či úpravu uživatele. Že přiřazení funguje správně bylo zkontrolováno na hlavní straně, při kliknutí na daného uživatele. Ve spodní straně tohoto formuláře se vypíší jak osobní informace uživatele, tak jeho přiřazená práva. Po otevření formuláře pro správu oprávnění bylo testováno přidávání, úprava a mazání jednolivých práv. Mazání práv a také uživatelů muselo být také ošetřeno vyjímkami. Uživatel nelze vymazat, pokud má přiřazená práva a také pokud provedl s pacientem nějaké vyšetření. Oprávnění nelze vymazat, pokud je přiřazeno k nějakému uživateli.

Tabulka 16: Testovací protokol administrátorského modulu

Administrátorský modul	
PŘIHLÁŠENÍ	
Správné přihlášení	Otevře se aplikace
Špatné přihlášení	Chybová hláška
HLAVNÍ STRANA	
Vyhledání uživatele	Filtrace seznamu dle jméno nebo uživatelského jména
Informace o uživateli	Kliknutím na uživatele ze seznamu se zobrazí jeho údaje a přidělené role
NOVÝ UŽIVATEL	
Tlačítko nový uživatel	Otevře se formulář pro přidání nového uživatele a zobrazí se všechny dostupné role
Uložení nového uživatele	Nový uživatel se tlačítkem Přidat uloží do databáze a přiřadí se mu vybrané role. Formulář se zavře a seznam se aktualizuje.
SMAZAT UŽIVATELE	
Tlačítko smazat uživatele	Smaže se uživatel z databáze a aktualizuje se seznam
Mazání uživatele s rolí nebo proběhlým vyšetřením	Chybová hláška
EDITOVAT UŽIVATELE	
Tlačítko Editovat uživatele	Vybráním uživatele a stisknutím tlačítka se otevře okno s jeho údaji a přiřazenými rolemi
Uložení změn uživatele	Provedením změny a stisknutím tlačítka Upravit se změny uloží do databáze a seznam se aktualizuje
SPRÁVA OPRÁVNĚNÍ	
Tlačítko Správa oprávnění	Stisknutím daného tlačítka se otevře okno pro správu rolí
Přidat novou roli	Stisknutím daného tlačítka se otevře okno pro přidání nové role
Uložení nové role	Stisknutím tlačítka Uložit se nová role uloží do databáze, okno se zavře a seznam rolí se aktualizuje.
Nevyplním novou roli a stisknu uložit	Chybová hláška
Upravit roli	Vybráním role a stisknutím daného tlačítka se otevře okno pro úpravu dané role.
Uložení změn role	Uložením se změna zapíše do databáze a seznam rolí se aktualizuje
Nevyberu žádnou roli a stisknu Upravit	Chybová hláška
Tlačítko Storno	Zavře se okno pro úpravy
Mazání role	Vyberu roli a ta se smaže ze seznamu i z databáze
Mazání role, který se používá	Chybová hláška
Nevyberu žádnou roli a stisknu Odebrat	Chybová hláška
O APLIKACI	
Tlačítko O aplikaci	Na hlavní straně stisknu dané tlačítko a otevře se okno s informacemi o aplikaci
KONEC APLIKACE	
Tlačítko konec	Stisknutím tlačítka konec se celá aplikace ukončí

Závěr

Výsledkem mé práce je navržení databáze pacientů a uživatelů oční kliniky, která bude sloužit jako jednotná databáze pro více aplikací vyvíjených na oční klinice ve Fakultní nemocnici v Ostravě. Databáze a vytvořené moduly jsou připraveny také pro rozšíření na ostatní oddělení nemocnice.

Věnovala jsem velkou pozornost analýze současného chodu na oční klinice a v prvním bodě teoretické části, jsem se věnovala popisu Oční kliniky. Jsou zde uvedeny ambulance, které se na klinice využívají a je zde popsán jejich význam a způsob vyšetřování pacientů. Na klinice se nachází všeobecná, glaukomová a vitreoretinální poradna, dále centrum pro léčbu VPMD a centrum pro léčbu dětí s vadami zraku. V každé ambulanci se provádí jiná vyšetření a je zapotřebí, aby lékař vyšetřující konkrétního pacienta měl k dispozici výsledek z každého vyšetření, které pacient provedl.

V dalších bodech teoretické části jsem se zabýva možnostmi jazyka C#. Je zde uveden postup pro vytvoření Windows aplikací v jazyce C#. v prostředí Visual Studia .NET. Velkou roli v této diplomové práci hraje také jazyk SQL, který je zde použit pro tvorbu a komunikaci s databází. Samotná tvorba databáze a přehled databázových prostředků je uveden v dalším bodě teoretické části, kde jsou popsány především relační databáze a je zde věnována pozornost databázovým prostředkům, především databázím Firebird.

Praktická část začíná analýzou požadavků z Oční kliniky. Úkolem bylo vytvořit jednotnou databázi, modul pro správu pacientů a administrátorský modul. Prvním bodem byl návrh databáze, která byla vytvořena v databázovém systému Firebird v grafickém prostředí IbExpert. Byly zde vytvořeny entity potřebné pro činnost modulu pro správu pacientů a administrátorského modulu, ale také pro potřeby vyvíjené aplikace pro testování barvocitu a zrakové ostrosti.

Dále byly navrženy a implementovány oba již zmíněné moduly. Modul pro správu pacientů slouží pro přidávání, úpravu a mazání pacienta. Je zde možnost podívat se na proběhlá vyšetření u daného pacienta, jejich datum a parametry. Do této aplikace mají přístup sestry na přijmu oční kliniky a sestry v laboratoři. Zdravotní sestra na příjmu nového pacienta vloží do databáze, uloží se k němu jeho osobní data a budou se zde vkládat všechna vyšetření, které pacient absolvuje.

Další vyvinutou aplikací v této diplomové práci je administrátorský modul, který slouží pro vkládání, úpravu či mazání uživatelů pracujících na oční klinice. Tento modul obsluhuje pouze administrátor, má možnost také vkládát oprávnění ke konkrétním aplikacím a přiřazovat je daným uživatelům. Těm také stanovuje uživatelské jméno a heslo, kterým se budou přihlašovat do potřebných aplikací, ke kterým mají přiděleno právo.

Databáze a jednotlivé moduly byly otestovány na Oční klinice ve Fakultní nemocnici v Ostravě. Bylo zde založeno 30 pacientů, kteří po uložení jejich údajů do databáze provedli vyšetření barvocitu, které bylo svázáno s touto databází.

Použitá literatura

[1] CECELJA, Franjo. *Manufacturing information: analysis, design*. London: Penton, 2002. ISBN 18-571-8031-3.

[2] ROBINSON, Simon. *C#: programujeme profesionálně*. Brno: Computer Press, 2003. Programmer to programmer. ISBN 80-251-0085-5.

[3] GROFF, James R. a Paul N. WEINBERG. *SQL: kompletní průvodce*. Brno: CP Books, 2005. Programování (CP Books). ISBN 80-251-0369-2.

[4] PRICE, Jason. *C#: programování databází*. Praha: Grada, 2005. Profesionál. ISBN 80-247-0982-1.

[5] MOMJIAN, Bruce. *PostgreSQL: praktický průvodce*. Brno: Computer Press, 2003. ISBN 80-7226-954-2.

[6] CÍSAŘ, Pavel. *InterBase/FireBird: podrobná příručka : tvorba, programování a správa databází*. Brno: Computer Press, 2003. ISBN 80-7226-946-1.

[7] *ITnetwork.cz* [online]. Praha: David Čápka, 2014 [cit. 2018-04-18]. Dostupné z: https://www.itnetwork.cz

[8] *DotNETportal.cz* [online]. Tomáš Herceg, Tomáš Jecha, 2013 [cit. 2018-04-19]. Dostupné z: https://www.dotnetportal.cz/

[9] VIRIUS, Miroslav. *C#: hotová řešení*. Brno: Computer Press, 2006. K okamžitému použití (Computer Press). ISBN 80-251-1084-2.

[10] *Webový vývoj v ASP.NET 2.0 pomocí bezplatných Express nástrojů pro úplné začátečníky* [online]. In: . Microsoft, 2005, s. 76 [cit. 2018-04-23]. Dostupné z: http://www.cs.vsb.cz/behalek/frvs/2005/dotnet/asp/ASP.NET.pdf

Seznam příloh

Příloha I Obsah CD

- Diplomová práce
- Zdrojový kód aplikace pro správu pacientů a administrátorského modulu